# Squeeze-and-Excitation Networks

E4040.2021Fall.FREN.report
Anh-Vu Nguyen an3078, Antonin Vidon av3023, Wael Boukhobza wab2138
*Columbia University*

## Abstract

*This report summarizes the findings of the original Squeeze-and-Excitation Networks paper and shows a reproduction of the results with Tensorflow [1][6]. Convolutional neural networks are a building block of all state-of-the-art models for image classification, including the well-known ResNet architecture. This report investigates the effectiveness of Squeeze-and-Excitation blocks whose aim is to strengthen the inter-channel relationship by performing channel-wise feature recalibration. The original paper showed that it consistently increased the classification accuracy with various data sets such as Image Net. This paper is going to depict how we were able to increase the accuracy of CNN models with the Squeeze-and-Excitation method. Because of technical barriers, we came up with slightly different models and data sets. We also tried to analyze the effect of the key parameters of Squeeze-and-Excitation blocks on our models, matching the paper's findings with varying success.*

## 1. Introduction

Convolutional neural networks (CNNs) are widely used for image classification as the filters allow to encode localized information and relations [2],[3]. Deep convolutional neural networks allowed increasing accuracy on data sets such as ImageNet [4]. Those networks use a series of downsampling operations and non-linear activation functions to generate powerful representations. There have been multiple attempts and propositions to increase the learning capability of CNNs by integrating blocks that encode local correlations between representations. Those local relationships can be spatial in the case of Inception networks [3][5], or between channels. The latter is what Squeeze-and-Excitation Networks aims to provide [6].

The original paper [6] shared the results of Squeeze-and-Excitation (SE) blocks on various network architectures (Resnet, VGG, Inception-ResNet-v2, BN-Inception, ResNeXt, and more), achieving consistent accuracy improvements over their non-SE counterparts. We tried to replicate those results on as many models as possible, but our limited computational capabilities (Tesla T4 vs 8 NVIDIA Titan X GPUs in the paper) meant that we had to make compromises; namely we used models with fewer parameters and smaller data sets. To keep training times under reasonable lengths, we created custom Resnet models, as well as custom ResNeXt and custom InceptionV3. Smaller data sets such as Tiny ImageNet also reduced training times but introduced other complications such as overfitting that made some analyses more difficult to reproduce.

More specifically, even though we were able to improve the classification accuracy with SE-blocks, some observations and tests on the effectiveness of SE parameters did not entirely match the original findings, and we will attempt to explain them in light of the changes that we made. Those tests concern SE-block variants, and activation weight analysis introduced by the paper. More detail will be provided in sections 3 and 5.

## 2. Summary of the Original Paper

### 2.1 Methodology of the Original Paper

Squeeze-and-Excitation Networks [6] introduce an architectural unit designed to capture cross-channel correlations and hence, improve the power of representation of CNNs. Instead of giving the same credit to all input channels by adding them altogether after convolution, SE blocks adaptively learn how to optimally weigh each feature map. Combined with a convolutional operator, these modules are not only capable of finding spatial features but also determine what channel is more relevant given the context. To achieve such a level of understanding, an SE block consists of three core components that successively perform the following operations: *squeeze*, *excite* and *scale*.

The *squeeze* module (**Fig. 2**) first acquires a global understanding of channels by converting each of them to a single numerical value through average pooling. This essentially builds a feature descriptor from which the block will learn how to extract the importance of the input filters.

With all $C$ channels converted to this single sequence of numerical values, the *excitation* module comes next as the adaptive component of the unit. A bottleneck MLP with one hidden layer encodes the feature descriptor in a space of lower dimension defined by a reduction factor $r$. Then, it maps back the downsampled sequence of dimension $C/r$ to an output layer with as many neurons as there were channels in the original input.

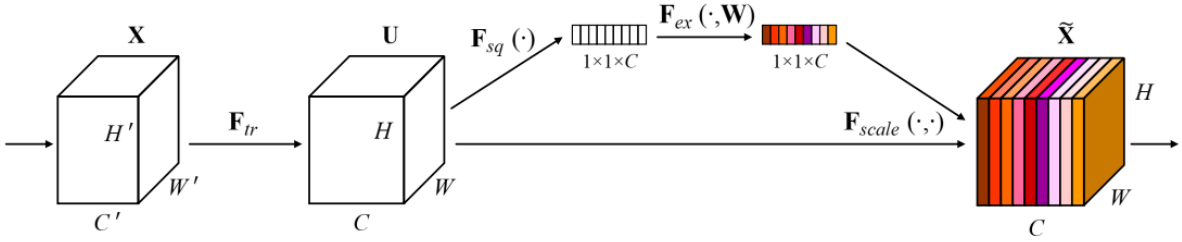



Fig. 1. A Squeeze-and-Excitation block.

Finally, the "excited" tensor goes through the *scale* module. After a sigmoid activation layer which computes the weight of each channel, each feature map of the original input is multiplied element-wise by the corresponding factor. The output of this transformation is the adaptively scaled version of the initial tensor.

The whole SE block can thus be summarized as followed using notations of **Fig. 1**:

1. The SE block goes after the convolutional transformation $F_{tr}$ that maps an input $X$ to a feature map $U = [u_1, u_2, ..., u_c]$ containing $C$ channels ($H$ and $W$ are the input dimensions).
2. The global average pooling layer allows to shrink the feature map $U$ to $C$ scalar channel descriptors $z = F_{sq}(U) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i,j)$.
3. Then the channel descriptors $z$ is fed into the bottleneck MLP $F_{ex}$ with two fully connected layers whose dimension depend on the ratio r **(Fig. 2)**, with a relu δ and sigmoid σ activation: $s = F_{ex}(z, W) = \sigma(W_{fc2}\delta(W_{fc1} * z))$.
4. The activated output $s$ is a sequence of $C$ scalars that are going to be used to rescale the convolutional transformation output $U$: $U_{rescaled} = F_{scale}(U, s)$, where $F_{scale}$ refers to an element-wise multiplication between the channels in $U$ and the elements of the vector $s$.

The Squeeze-and-Excitation module is used to build variants of state-of-the-art CNN-based models for image classification. The main novel architecture introduced in the paper is an upgraded version of ResNet in which an SE block has been plugged after each residual block of the original architecture. The transformation is similar for ResNeXt architectures.

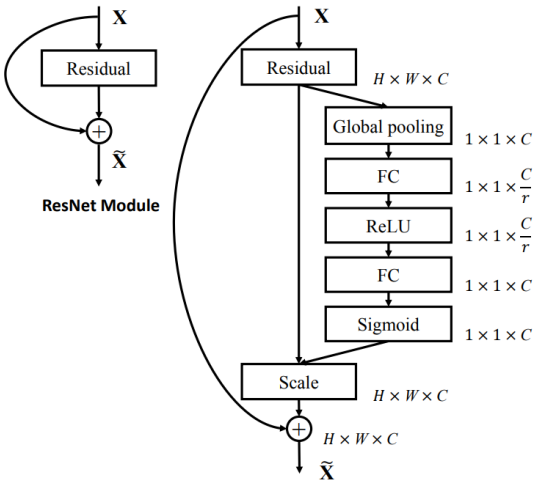



***Fig. 2*** *The schema of the original Residual module (left) and the SE-ResNet module (right)*

For the SE-ResNet implementation, the paper suggested other integrations where SE-blocks were added at other places as shown in **Fig. 3**

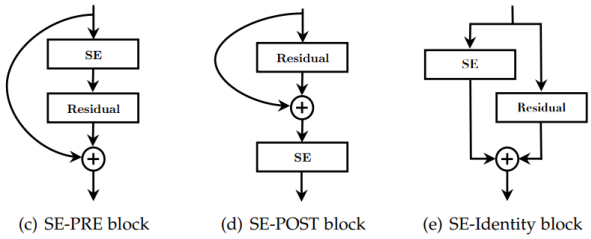



***Fig. 3*** *SE block integration designs. Standard SE block is shown in* ***Fig. 2***

More precisely, the SE-PRE block goes before the residual unit, while the SE-POST block is moved after the summation layer. The SE-Identity block is placed in the skip connection.

| | original | | re-implementation | | | SENet | | |
|---|---|---|---|---|---|---|---|---|
| | top-1 err. | top-5 err. | top-1 err. | top-5 err. | GFLOPs | top-1 err. | top-5 err. | GFLOPs |
| ResNet-50 [13] | 24.7 | 7.8 | 24.80 | 7.48 | 3.86 | $23.29_{(1.51)}$ | $6.62_{(0.86)}$ | 3.87 |
| ResNet-101 [13] | 23.6 | 7.1 | 23.17 | 6.52 | 7.58 | $22.38_{(0.79)}$ | $6.07_{(0.45)}$ | 7.60 |
| ResNet-152 [13] | 23.0 | 6.7 | 22.42 | 6.34 | 11.30 | $21.57_{(0.85)}$ | $5.73_{(0.61)}$ | 11.32 |
| ResNeXt-50 [19] | 22.2 | - | 22.11 | 5.90 | 4.24 | $21.10_{(1.01)}$ | $5.49_{(0.41)}$ | 4.25 |
| ResNeXt-101 [19] | 21.2 | 5.6 | 21.18 | 5.57 | 7.99 | $20.70_{(0.48)}$ | $5.01_{(0.56)}$ | 8.00 |
| VGG-16 [11] | - | - | 27.02 | 8.81 | 15.47 | $25.22_{(1.80)}$ | $7.70_{(1.11)}$ | 15.48 |
| BN-Inception [6] | 25.2 | 7.82 | 25.38 | 7.89 | 2.03 | $24.23_{(1.15)}$ | $7.14_{(0.75)}$ | 2.04 |
| Inception-ResNet-v2 [21] | $19.9^{\dagger}$ | $4.9^{\dagger}$ | 20.37 | 5.21 | 11.75 | $19.80_{(0.57)}$ | $4.79_{(0.42)}$ | 11.76 |

***Fig. 4*** *Classification errors with ImageNet. Error gains with SEnets is shown within the parentheses.*

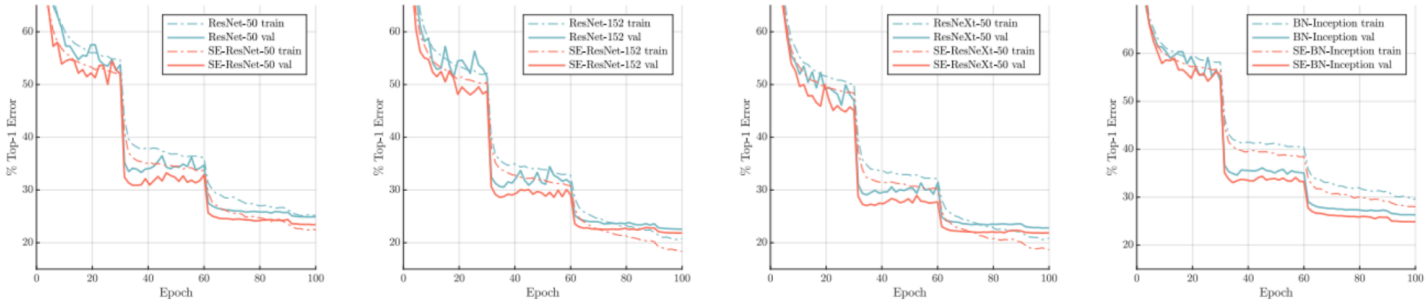



***Fig. 5*** *Training and validation errors*

The authors also propose an amelioration of the Inception architecture for which an SE block has been plugged just after each Inception block.

To train the SE-nets with ImageNet, the paper uses the SGD optimizer with momentum and a decreasing learning rate, divided by 10 after 30 epochs. They used a minibatch size of 1024, set the initial learning rate to 0.6, epochs to 100, and the SE ratio $r$ to 16. The training strategy is completed by a mild data augmentation used with horizontal flipping and random cropping.

## 2.2 Key Results of the Original Paper

### 2.2.1 Accuracy results

The paper showed that SE implementations resulted in measurable accuracy improvements (**Fig. 4**). For instance, the SE-ResNet-50 model had a 0.86% top-5 accuracy improvement over the ResNet-50 one, bridging the gap between ResNet-101 that has twice as many parameters ($44$ vs $23 \times 10^6$ ). On the other hand, only 2.5 million new parameters were introduced by adding the SE blocks. As illustrated, these modules give the capability to match much deeper networks in terms of accuracy while only adding a small amount of parameters. Similarly to SE-Resnet-50, SE-ResNet-101 yields better results than ResNet-152 by 0.27% on the ImageNet dataset.

The paper also obtained similar results on other datasets such as CIFAR-10 and CIFAR-100, with SE implementations surpassing deeper network accuracy.

Classification error (%) on CIFAR-10.

| | original | SENet |
|---|---|---|
| ResNet-110 [14] | 6.37 | 5.21 |
| ResNet-164 [14] | 5.46 | 4.39 |
| WRN-16-8 [67] | 4.27 | 3.88 |
| Shake-Shake 26 2x96d [68] + Cutout [69] | 2.56 | 2.12 |

TABLE 5
Classification error (%) on CIFAR-100.

| | original | SENet |
|---|---|---|
| ResNet-110 [14] | 26.88 | 23.85 |
| ResNet-164 [14] | 24.33 | 21.31 |
| WRN-16-8 [67] | 20.43 | 19.14 |
| Shake-Even 29 2x4x64d [68] + Cutout [69] | 15.85 | 15.41 |

***Fig. 6*** *Classification results on the CIFAR-10 and CIFAR-100 datasets*

### 2.2.2 Testing the ratio value

The paper then tested the effect of various SE parameters on the behavior of SE-based networks. It was shown that different ratio values impacted the generalization capability of the models, and hence the test

accuracy. There is thus a preference for ratios equal to 16 as they do not add as many parameters as lower ratios do, and provide the second-best accuracy improvement over the tested values.

Single-crop error rates (%) on ImageNet and parameter sizes for SE-ResNet-50 at different reduction ratios. Here, *original* refers to ResNet-50.

| Ratio $r$ | top-1 err. | top-5 err. | Params |
|---|---|---|---|
| 2 | 22.29 | 6.00 | 45.7M |
| 4 | 22.25 | 6.09 | 35.7M |
| 8 | 22.26 | 5.99 | 30.7M |
| 16 | 22.28 | 6.03 | 28.1M |
| 32 | 22.72 | 6.20 | 26.9M |
| original | 23.30 | 6.55 | 25.6M |

***Fig.7*** *Classification results with different ratios*

### 2.2.3 Ablation study

The study of SE integration on different layers of ResNet models showed that the performance of SE networks can be improved by carefully choosing where to insert the SE blocks. SE-POST and the standard SE integrations showed better accuracy than the SE-PRE and SE-Identity ones, with a 0.58% of top-1 accuracy delta between the best performing implementation and the worst implementation.

Effect of different SE block integration strategies with ResNet-50 on ImageNet (error rates %).

| Design | top-1 err. | top-5 err. |
|---|---|---|
| SE | 22.28 | 6.03 |
| SE-PRE | 22.23 | 6.00 |
| SE-POST | 22.78 | 6.35 |
| SE-Identity | 22.20 | 6.15 |

***Fig. 8*** *Classification results with different SE-block integrations*

### 2.2.4 Stage integration study

The standard SE version of ResNet models tested by the paper had SE blocks in all stages. By only adding SE layers at specific stages, the paper showed that SE blocks have different effects depending on their position in the architecture.

Effect of integrating SE blocks with ResNet-50 at different stages on ImageNet (error rates %).

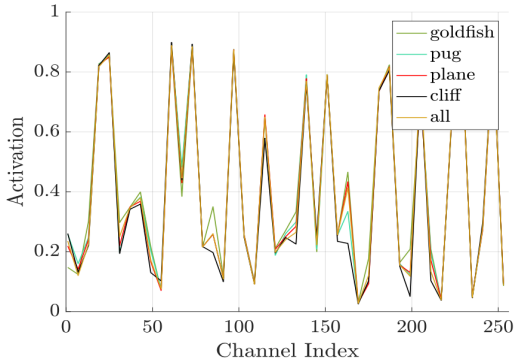
| Stage | top-1 err. | top-5 err. | GFLOPs | Params |
|---|---|---|---|---|
| ResNet-50 | 23.30 | 6.55 | 3.86 | 25.6M |
| SE_Stage_2 | 23.03 | 6.48 | 3.86 | 25.6M |
| SE_Stage_3 | 23.04 | 6.32 | 3.86 | 25.7M |
| SE_Stage_4 | 22.68 | 6.22 | 3.86 | 26.4M |
| SE_All | 22.28 | 6.03 | 3.87 | 28.1M |

***Fig. 9*** *Classification results with different SE-blocks added at different stages in the ResNet-50 model*

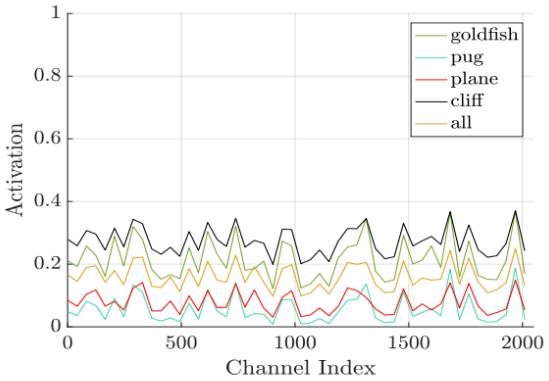
While no major conclusion can be drawn on the superiority of certain stage SE integrations, they all positively enhance the accuracy of the ResNet-50 model, and it appears that the gains are combined together in the SE_all version with the latest stage yielding the best improvements.

### 2.2.5 Activation weights analysis

The study of activation distributions revealed that they are more class-dependent on the later stages (last layers).




(a) SE_2_3

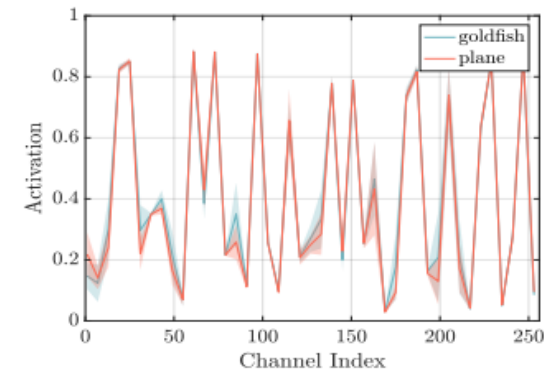



(f) SE_5_3

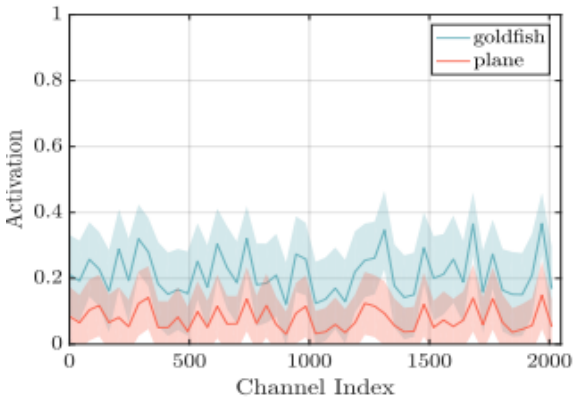
***Fig. 10*** *Activation values for each channel. Location in the ResNet-50 model is given by SE_stageID_blockID.*

The activation distributions are indeed separable for different classes on the latest stages. This also suggests that more important channel relations are less related to classes in earlier stages.

Finally, a tendency to diversify the representation of a class in the deeper stages was observed. This can be seen by the increased standard deviation of the distributions.




(a) SE_2_3




(f) SE_5_3

***Fig. 11*** *Activation values and standard deviation for each channel. Location in the ResNet-50 model is given by SE_stageID_blockID*

### 2.2.6 Inference time

The original paper reported a 10% increase in inference time due to the addition of SE layers. Predictions for a mini-batch of 256 samples took 209 ms on SE-ResNet-50 compared to 190 ms on ResNet-50.

## 3. Methodology (of the Students' Project)

Although we tried to replicate the paper's methodology, because of various factors, some modifications had to be made to make the testing feasible.

### 3.1. Objectives and Technical Challenges

The objectives of this report are to reproduce the architecture of SE-ResNet, SE-ResNeXt, and SE-Inception models, and train them using data sets that are as close as possible to the ones used by the paper.

We are also going to perform the ablation, ratio, stages and activation studies as the original authors did. Overall, the technical challenges were:

**Work within computational constraints:** the original paper [6] mentioned the use of 8 Nvidia Titan X GPUs for a combined 53.528TFLOPS of single precision FP32 performance and 96GB of VRAM. We only had access to a Nvidia T4 GPU from the Google Cloud Platform granting 8.1TFLOPS of single precision FP32 performance and 16GB of VRAM. Despite the newer hardware, we still had to work with significantly less resources and thus had to carefully choose the models, data sets and hyperparameters to keep the training times within reasonable lengths. Smaller models take less space in memory and allow bigger batch sizes but decrease the capacity. Smaller data sets take less time to train and do not have to be loaded from the drive during training, but are more prone to variability and overfitting.

**Build SE models:** independent SE blocks may not be hard to design in Tensorflow 2.4, but integrating it into existing models such as ResNet-50 will require careful examination of all parameters. Tensorflow has some models that can be imported but not all are available, and building them from scratch might be the better approach. Fortunately, ample resources and papers [2] allowed us to overcome this difficulty and we were able to build custom models tailored to our needs.

### 3.2.Problem Formulation and Design Description

To deal with the computational disadvantage, we chose to work on different but similar models. Instead of working with ResNet-50, ResNet-101 and ResNet152 models, we choose to work with ResNet-18, ResNet-34 and ResNet-50 models. The latter one being one of the biggest models that we trained : 24 million parameters. Overall our ResNet models had from 11 million to 24 million parameters, the SE variants having up to 26.5 million parameters. To expand our range of models, we also implemented the more recent ResNeXt, which introduces grouped convolutions in a ResNet-like architecture. As ResNeXt architectures are typically more challenging to train than ResNet, we only focused on ResNeXt-29 (8 x 64d) , consisting of 29 layers and 8 grouped convolutions for every stage, for a total of 34 million parameters. We also implemented the InceptionV3 model, which is 48 layers deep and has 24.5 million parameters. The SE variant has 26.2 million parameters.

To keep training times under reasonable lengths, we replaced ImageNet (1.35 million images) with Tiny

ImageNet (100k images across 200 classes) [11], reducing the sample resolution from 224x224 to 64x64. This change allowed us to keep training times under 2.5 hours with the chosen hyperparameters. Unfortunately, the smaller data set also introduced overfitting, which was somewhat reduced with data augmentation.

| output | **18-layer** | **34-layer** | **50-layer** |
|---|---|---|---|
| 64 × 64 | Input | | |
| 32 × 32 | 7 × 7, 64, stride2 | | |
| 16 × 16 | 3 × 3max pool, stride 2 | | |
| | [3 × 3, 64]<br>[3 × 3, 64]<br>× 2 | [3 × 3, 64]<br>[3 × 3, 64]<br>× 3 | [1 × 1, 64]<br>[3 × 3, 64]<br>[1 × 1, 256]<br>× 3 |
| 8 × 8 | [3 × 3, 128]<br>[3 × 3, 128]<br>× 2 | [3 × 3, 128]<br>[3 × 3, 128]<br>× 4 | [1 × 1, 128]<br>[3 × 3, 128]<br>[1 × 1, 512]<br>× 4 |
| 4 × 4 | [3 × 3, 256]<br>[3 × 3, 256]<br>× 2 | [3 × 3, 256]<br>[3 × 3, 256]<br>× 6 | [1 × 1, 256]<br>[3 × 3, 256]<br>[1 × 1, 1024]<br>× 6 |
| 2 × 2 | [3 × 3, 512]<br>[3 × 3, 512]<br>× 2 | [3 × 3, 512]<br>[3 × 3, 512]<br>× 3 | [1 × 1, 512]<br>[3 × 3, 512]<br>[1 × 1, 2048]<br>× 3 |
| 1 × 1 | average pool, 200-d fc, softmax | | |
| Params | $12.7 \times 10^6$ | $22.8 \times 10^6$ | $23.9 \times 10^6$ |

***Fig. 12*** *ResNet models descriptions for Tiny ImageNet*

We used the same SGD optimizer as in the paper, and changed the learning rate reduction strategy and epochs to suit our models and data. More detail can be found in section 4.

To build our models and their SE counterparts, we developed custom scripts that could build ResNet and ResNeXt models with any specified kernel sizes, depth and parameters. This allowed us to come up with a custom ResNet model which we called ResNet-46 that was more appropriate for smaller images: regular ResNet models [2] can only accept images with resolution larger than 32x32 pixels. However at the lower limit of 32x32, the last convolutional layers only produce feature channels that are of dimension 1.

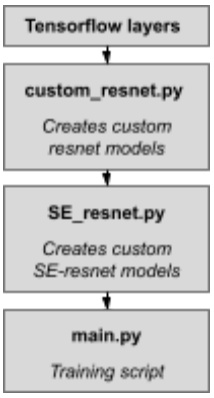



***Fig. 13*** *Simplified code dependency diagram for Resnet models*

| Stage | output | **34-layer** | **46-layer** |
|---|---|---|---|
| | 32 × 32 | Input | |
| 1 | 16 × 16 | 7 × 7, 64, stride2 | |
| 2 | 8 × 8 | 3 × 3max pool, stride 2 | |
| | | [3 × 3, 64]<br>[3 × 3, 64]<br>× 3 | [3 × 3, 64]<br>[3 × 3, 64]<br>× 4 |
| 3 | 4 × 4 | [3 × 3, 128]<br>[3 × 3, 128]<br>× 4 | [3 × 3, 128]<br>[3 × 3, 128]<br>× 6 |
| 4 | 2 × 2 | [3 × 3, 256]<br>[3 × 3, 256] | [1 × 1, 256]<br>[3 × 3, 256]<br>[1 × 1, 1024]<br>× 8 |
| 5 | 1 × 1 | [3 × 3, 512]<br>[3 × 3, 512]<br>× 3 | x |
| | 1 × 1 | average pool, 10 (or 100)-d fc, softmax | |
| Params | | $\sim 22 \times 10^6$ | $\sim 11 \times 10^6$ |

***Fig. 14*** *ResNet-46 description for CIFAR images with comparison to ResNet-34*

| Stage | output | **ResNeXt-29, 8 x 64d** |
|---|---|---|
| | $32 \times 32$ | Input |
| 1 | $16 \times 16$ | $3 \times 3$, 64, stride 2 |
| 2 | $8 \times 8$ | $3 \times 3$ max pool, stride 2 |
| | | $[1 \times 1, 512]$<br>$[3 \times 3, 512]$, # groups = 8<br>$[1 \times 1, 256]$<br>$\times 3$ |
| 3 | $4 \times 4$ | $[1 \times 1, 1024]$<br>$[3 \times 3, 1024]$, # groups = 8<br>$[1 \times 1, 512]$<br>$\times 3$ |
| 4 | $2 \times 2$ | $[1 \times 1, 2048]$<br>$[3 \times 3, 2048]$, # groups = 8<br>$[1 \times 1, 1024]$<br>$\times 3$ |
| | $1 \times 1$ | average pool, 10 (or 100)-d fc, softmax |
| Params | | $\sim 34.4 \times 10^6$ |

***Fig. 15*** *ResNeXt-29 (8 x 64d) description for CIFAR images*

In addition to the Tiny ImageNet data set, we also used the CIFAR-10 and CIFAR-100 for their small size despite having 60k samples of resolution 32x32. This is the reason why we opted to create ResNet46.

Because we obtained better result consistency, especially with CIFAR-10 we decided to conduct all ablation (**Fig. 3**), ratio (**Fig. 16**), stages (**Fig. 14**) and activation studies with the CIFAR-10 data set.

The activation values for each channel is the output of the second fully connected layer in the SE block $s = F_{ex}(z, W) = \sigma(W_{fc2}\delta(W_{fc1} * z))$ as described in section 2.1. We want to study the distribution across classes for different stages.

More information on the specific implementations are given in the next section.

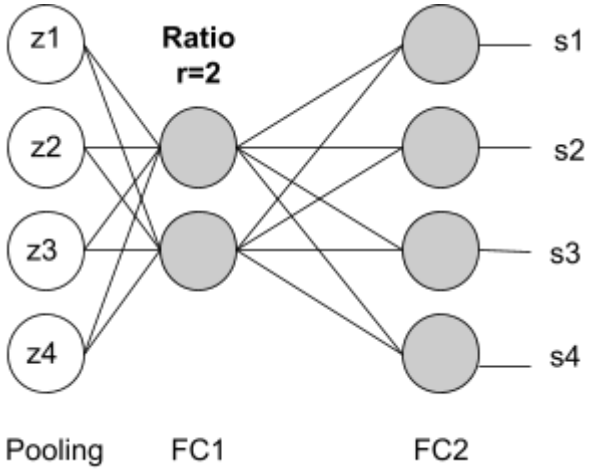



***Fig. 16*** *Example of fully connected layers in a SE-Block with 4 filter channels and a ratio of 2*

## 4. Implementation

In this section, we describe the framework we used to demonstrate the contribution of SE blocks to achieving better classification performance. In particular, we delineate the choice of architecture we made for our models, specific to the selected datasets.

### 4.1 Data

**Data description:** In order to assess the performance of our models with and without correlation modules, we used state-of-the-art multi-label image datasets. We picked CIFAR-10 [9] and CIFAR-100 [10] as they were part of the original paper. We also picked Tiny ImageNet [11] as a lighter version of ImageNet [12], widely used for benchmarking. We chose not to include the latter as this one was beyond the scope of our computational capabilities.

The CIFAR data sets could be imported right from Tensorflow into testing and training data sets. To ensure a pseudo uniform class distribution, the training samples were shuffled. We split the training data to get 2000 validation samples and 48000 training samples.

| **Dataset name** | **# images** | **# classes** | **shape** |
|---|---|---|---|
| CIFAR-10 [9] | 60,000 | 10 | 32x32x3 |
| CIFAR-100 [10] | 60,000 | 100 | 32x32x3 |
| Tiny ImageNet [11] | 100,000 | 200 | 64x64x3 |

***Fig. 17*** *Datasets used with their number of samples, classes and shapes*

The Tiny ImageNet was downloaded from the web: http://cs231n.stanford.edu/tiny-imagenet-200.zip . We then used the *Image* function of the *PIL* package to read the images into numpy arrays.

**Data preprocessing:** There are 500 training images for each class in the Tiny ImageNet, however after discarding those who only had one channel instead of three, each class had a remaining average of 490 samples. The Tiny ImageNet was intended to be used for a challenge, and as such does not have labeled testing data. This means that the test samples could not be used. We decided to use the provided validation data for our testing data, and then split the training data to generate new validation and training sets (**Fig. 17**).

| **Dataset name** | # training samples | # validation samples | # Testing samples |
|---|---|---|---|
| CIFAR-10 | 48 000 | 2 000 (4.2%) | 10 000 |
| CIFAR-100 | 48 000 | 2 000 (4.2%) | 10 000 |
| Tiny ImageNet | 93 179 | 9 832 (10.6 %) | 5 000 |

***Fig. 18*** *Testing, validation and training data sets*

We used data augmentation to reduce overfitting, while not completely eliminating it on the Tiny ImageNet. However it is interesting to see how SE blocks behave when the model is prone to overfitting. For the CIFAR-10 and 100 training we used data augmentation suggested by the authors: 4 pixels width shift, 4 pixels height shift and horizontal flip. Data augmentation is being applied with randomized parameters with the data generator preprocessing method from Tensorflow. For the Tiny Imagenet, we doubled the pixel shifts to account for the increased resolution, and added brightness change and rotations for more variability. The data augmentation effects are illustrated below in **Fig. 19**:

***Fig. 19*** *Example of data augmentation for TinyImage Net. Notice the random shifts*

Nonetheless, we did not use data augmentation for InceptionV3 models as we already needed to rescale the CIFAR images from 32x32 to at least 100x100 for these models (else the image would totally disappear at the end of the forward loop, due to the pooling operations). Indeed, we found that adding data augmentation on top of the resizing operation would totally break the structure of the images and make them unreadable for the algorithm, thus performing very badly on untransformed data.

## 4.1 Deep Learning Network

**Network architectures:** We used ResNet and ResNeXt models as described in **Fig. 12, Fig. 14** and **Fig. 15**. The SE versions have the same overall layout but the residual block is changed as described in **Fig. 20**.
The SE-blocks are integrated in the residual blocks and go after the last convolutional and batch normalization layer for each stage id and block id.

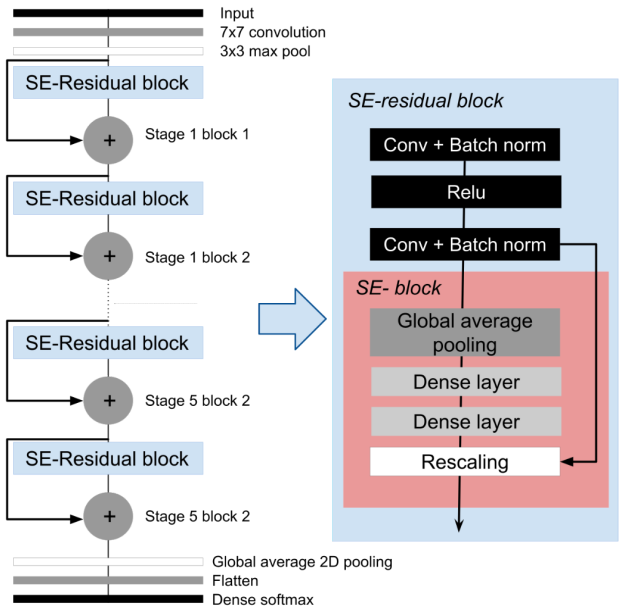



***Fig. 20*** *Architecture flow chart of SE-Resnet-18, with 3 layers*

Throughout all the testing, the following models were used: ResNet-18, ResNet-34, ResNet-50, our custom ResNet-46, ResNeXt29, InceptionV3 and the respective SE-net versions. Tiny ImageNet was used to train ResNet-18, ResNet-34, ResNet-50 and their SE variants, while CIFAR-10 and CIFAR-100 were used for ResNet-46, InceptionV3, ResNeXt29 models and their SE implementations.

**Parameter:** We trained our models by following as close as possible the choice of parameters of the original paper. The same SGD optimizer with momentum

set to 0.9 was used. For the Tiny ImageNet data set, the learning rate was set to 0.6 to match the paper and was divided by 10 after 20 epochs (instead of 30 in the paper) for a total of 60 epochs (instead of 100). The changes were made because the small dataset meant that the model learnt faster. A consequence of this is that overfitting occurred quite early (**Fig. 27**), which made us add early stopping with a patience of 15 epochs, to avoid unnecessarily long training times. The model with the best validation accuracy at each epoch was saved. We used a batch size of 128. Nonetheless, it is to note that due to the custom blocks present in Inception V3, it was not possible to save the whole model for this architecture, we were only able to save its weights at the end of our training.

The same hyper-parameters were used to train the CIFAR data sets, but we lowered the initial learning rate to 0.01 as the models had issues improving the accuracy in the initial iterations.

## 4.2 Software Design

**ResNet and SE-ResNet models:** We built ResNet models from scratch by creating a custom function that takes the overall structure of the desired ResNet model and outputs the corresponding Tensorflow model (**Fig. 21**). The function takes the list of kernel sizes, filters, and block multiplicity, as well as the input and output parameters. Link to the function.

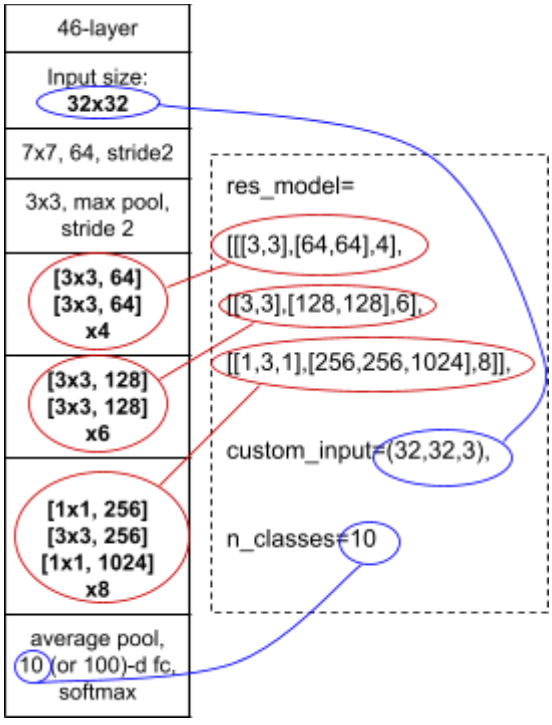



*Fig. 21 Description of the parameters of our custom resnet function generator*

The input size and output shape can be changed to accomodate for the different data set shapes and classes.

We then used that function to create SE versions of ResNet models (**Fig. 13**). Link to the function.
To create SE-ResNets, the function takes the lists of layers whose output is going to be fed into SE blocks. We then used the graph structure of our ResNet models to integrate SE blocks during reconstruction. SE blocks were placed after the batch normalization layers that followed the last convolutional layer of the residual units (**Fig.2**).

```
['conv2_block1_2_bn',
 'conv2_block2_2_bn',
 'conv3_block1_2_bn',
 'conv3_block2_2_bn',
 'conv4_block1_2_bn',
 'conv4_block2_2_bn',
 'conv5_block1_2_bn',
 'conv5_block2_2_bn']
```

*Fig. 22 Layers where SE blocks are going to be inserted in ResNet-18. Location in the ResNet-18 model is given by conv_stageID_blockID.*

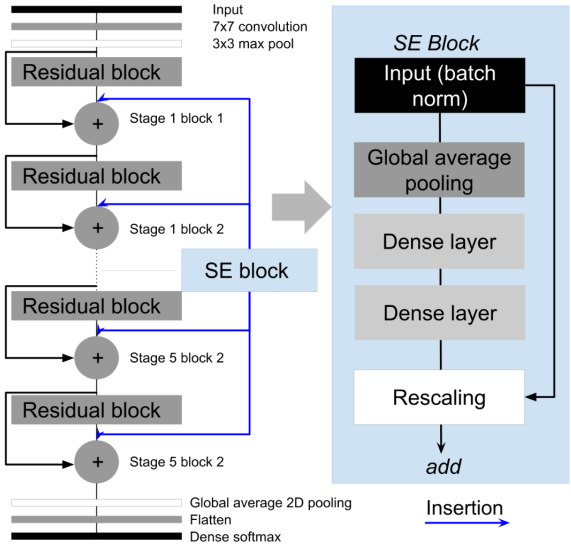



*Fig. 23 Architecture flow chart of SE-Resnet-18, indicating where SE blocks are added*

**Github link**:
https://github.com/ecbme4040/e4040-2021fall-project-fren-an3078-wab2138-av3023

**InceptionV3 and SE-InceptionV3 models:** Inception models are based on the idea of implementing a new range of blocks, called Inception blocks, in which we perform several convolutions in parallel for the same

input and then concatenate them. InceptionV3, compared to its predecessors, adds the possibility to break a same convolution from these inception blocks into two sequential convolutions of smaller kernels, performing similar results with fewer parameters.
We manually implemented this particular network, and designed the SE version of it according to the recommendations of the Squeeze-Excitation paper, i.e by adding SE blocks after each convolution block.

This model also has an auxiliary classifier that produces a softmax before the actual deep-end of the architecture. It only helps during the backward propagation phase of the training and acts as a regularizer according to the authors of the original paper of InceptionV3[14].

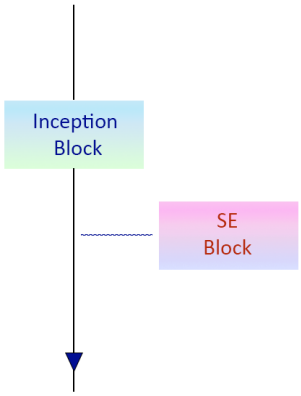



***Fig. 24*** *SE blocks are added after Inception blocks in InceptionV3.*

# 5. Results

## 5.1 Project Results

### 5.1.1 Accuracy results

To quantitatively assess the contribution of SE blocks to ResNet, ResNeXt and Inception architectures, we started with experiments on the lightest and most understood dataset at our disposal : CIFAR-10. We report the classification errors we obtained in **Fig. 25**.

As we hoped to observe, plugging SE blocks within convolutional stages of the models consistently led to an increase in classification performance. To be more specific, we see a 1.9% increase in classification error for ResNet46 and 0.2% increase for ResNeXt29. Top-3 and top-5 errors also testify for a superiority of the SE implementations.

| | **top-1 err.** | **top-3 err.** | **top-5 err.** |
|---|---|---|---|
| **ResNet46** | 19.30 | 4.14 | 1.25 |
| **SE-ResNet46** | 17.39 | 3.34 | 1.11 |
| **ResNeXt29** | 17.13 | 3.48 | 0.91 |
| **SE-ResNeXt29** | 16.91 | 3.40 | 0.86 |
| **InceptionV3** | 16.25 | 3.54 | 0.80 |
| **SE-InceptionV3** | 22.75 | 5.29 | 1.59 |

***Fig. 25*** *Top errors (%) on CIFAR-10*

We further assess the capabilities of SE blocks on the classification task for CIFAR-100. The key metrics are reported in **Fig. 26**.

| | **top-1 err.** | **top-3 err.** | **top-5 err.** |
|---|---|---|---|
| **ResNet46** | 50.00 | 31.99 | 23.18 |
| **SE-ResNet46** | 45.83 | 27.97 | 20.84 |
| **ResNeXt29** | 45.27 | 25.97 | 19.05 |
| **SE-ResNeXt29** | 44.72 | 26.14 | 18.84 |

***Fig. 26*** *Top errors (%) on CIFAR-100*

The conclusion is similar to the one we presented for CIFAR-10. Across all experiments, architectures including SE blocks consistently overperformed their counterparts. Namely, SE-ResNet46 topped ResNet46 error rate by 4.2% when SE-ResNeXt29 subceeded ResNeXt29 error rate by 0.6%. An interesting observation is that, as for CIFAR-10, SE blocks do not seem as efficient for ResNeXt as they are for ResNet. This is due to the fact that grouped convolutions and aggregated transformations already implement an adaptive differential treatment of channels by group. However, since the size of the set of transformations, called “cardinality”, is very small in comparison to the number of filters across stages (8 vs 256, 512 or 1024), SE blocks still improve the performance of ResNeXt. Unfortunately, we were not able to test the results of InceptionV3 on other databases than CIFAR-10 due to a lack of time, nor could we overperform the results of the original model (coded from scratch by us) with the SE version for the same reasons. Yet, we still wanted to mention these

results as on CIFAR-10, InceptionV3 managed to beat every over Top-1 and Top-5 accuracy, and **Fig. 30** shows that the validation accuracy is still growing. We think that with more time and epochs to train the SE version of the model, we would have overperformed the original Inception model and thus every other model.

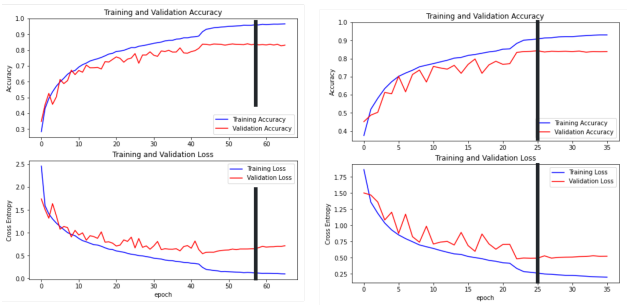



***Fig. 27*** *Training metrics for CIFAR-10. Left: ResNet-46. Right: SE-ResNet46. Black bars indicate when the best model was last saved*

It is also interesting to note that SE models generally had shorter training times, with the best model being saved at earlier epochs. For instance for the ResNet-46, the best model was saved at epoch 56 vs epoch 25 for the SE-ResNet-46. As a result, overfitting occurs faster for SE models.

Note that sometimes the validation accuracy increases while the loss function increases. This can be explained by the fact that some images are mislabeled [13] thus penalizing the loss function.

We next investigate if the benefits of SE blocks generalize to datasets beyond CIFAR-10 and 100. For this matter, we trained our modern architectures on Tiny ImageNet. Each baseline and its SENet counterpart is trained with or without the data augmentation strategy described in **4.1**. With this approach, we come up with more baselines to assess the contribution of SE blocks. We report the performance of all architectures in **Fig 28**. The figures in parenthesis materialize the reduction of error rate obtained after plugging the SE blocks into the model.

| | **Top-1 err.** | **Top-3 err.** | **Top-5 err.** |
|---|---|---|---|
| **ResNet18** | 71.69 | 56.24 | 48.2 |
| **SE-ResNet18** | 67.91 (3.78) | 51.97 (4.27) | 44.09 (4.11) |
| **ResNet18 w. DA** | 57.93 | 39.27 | 31.35 |
| **SE-ResNet18 w. DA** | 57.19 | 39.88 | 32.52 |

| | (0.74) | (-0.61) | (-1.17) |
|---|---|---|---|
| **ResNet34** | 71.45 | 54.45 | 44.9 |
| **SE-ResNet34** | 68.88 (2.57) | 51.99 (2.46) | 44.04 (0.86) |
| **ResNet34 w. DA** | 61.66 | 44.59 | 36.16 |
| **SE-ResNet34 w. DA** | 58.62 (3.04) | 40.94 (3.65) | 32.72 (3.44) |
| **ResNet50** | 76.19 | 60.03 | 51.09 |
| **SE-ResNet50** | 71.81 (4.38) | 55.51 (4.52) | 47.71 (3.38) |
| **ResNet50 w. DA** | 70.2 | 52.81 | 44.03 |
| **SE-ResNet50 w. DA** | 64.08 (6.12) | 46.02 (6.79) | 38.53 (5.5) |

***Fig. 28*** *Top errors (%) on Tiny ImageNet. SE improvement in parentheses*

With Tiny ImageNet, SE blocks yield a significant improvement in classification performance. On average, data augmentation improves the error rate by 10% and SE blocks further improve it by 3%. Overall, bigger architectures produce the worst outcomes because of their capacity but also benefit the most from modeling cross-channel correlation. In particular, ResNet50 classification performance improves by 4.4% without data augmentation and 6.1% with data augmentation.

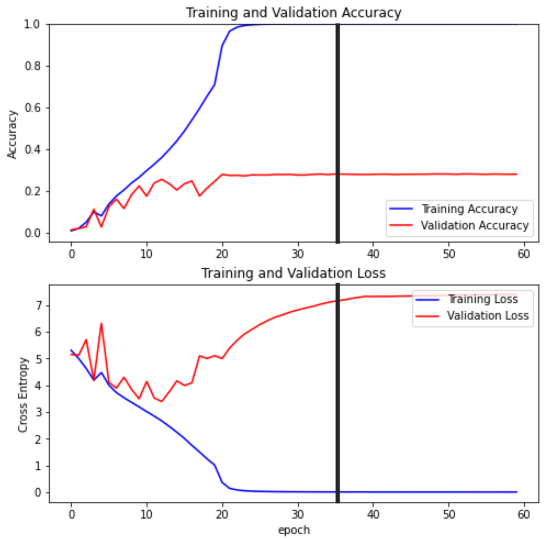



***Fig. 29*** *SE-ResNet-18 training and validation metrics. Black bars indicate when the best model was last saved*

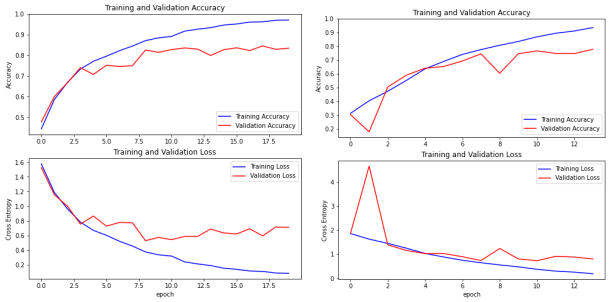



***Fig. 30*** *Training metrics for CIFAR-10 with InceptionV3. Left: InceptionV3. Right: SE-InceptionV3*

We had to save the best model at each epoch as all tested models had overfitting as in **Fig. 29**. It is however interesting to see that SE blocks can still improve the accuracy when limited training data is available.

### 5.1.2 Testing the ratio value

We tested the effect of the ratio parameter on the accuracy and number of parameters of the SE-ResNet-46 model with CIFAR-10.

| Ratio | Params | top-1 err. | top-3 err. | top-5 err. |
|---|---|---|---|---|
| **Resnet** | $11 \times 10^6$ | 19.30 | 4.14 | 1.25 |
| **1** | $28 \times 10^6$ | 17.52 | 3.86 | 0.88 |
| **2** | $19.5 \times 10^6$ | 17.33 | 3.54 | 0.9 |
| **4** | $15.2 \times 10^6$ | 18.43 | 3.9 | 1.25 |
| **8** | $13.1 \times 10^6$ | 17.68 | 3.84 | 1.04 |
| **16** | $12.1 \times 10^6$ | 17.39 | 3.34 | 1.01 |
| **32** | $11.5 \times 10^6$ | 17.94 | 3.67 | 1.05 |

***Fig. 31*** *SE-ResNet-46 test errors with different ratios*

All SE models yield accuracy improvements over the standard ResNet model. Despite having substantially more trainable parameters, SE models with lower ratio do not produce equally better accuracy. It seems that the increased capacity is offset by the better generalization capability of the bottleneck in higher ratios.

Differences in errors might come down to testing variability, and we would need to run those tests multiple times to come up with a clear conclusion on the effects of the ratio parameter.

### 5.1.3 Ablation study

Different SE setups were tested (**Fig. 3**).

| Implementation | top-1 err. | top-3 err. | top-5 err. |
|---|---|---|---|
| **Standard SE** | 17.39 | 3.34 | 1.11 |
| **SE-POST** | 21.08 | 4.6 | 1.11 |
| **SE-PRE** | 20.02 | 5.45 | 1.96 |
| **SE-Identity** | 43.78 | 15.48 | 6.03 |

***Fig. 32*** *SE-ResNet-46 test errors with different SE implementations*

The standard SE implementation had the best accuracy overall. We noticed that the other SE model versions had issues with a very low convergence with the same hyper parameters, however to keep the comparison consistent we decided not to change the way they were trained. This was particularly noticeable with the SE-Identity block that stabilized at a very low accuracy.

### 5.1.4 Stage integration study

We tested the effect of SE blocks at each stage by removing them on the other stages (**Fig. 33**).

| Stage | top-1 err. | top-3 err. | top-5 err. |
|---|---|---|---|
| **2** | 19.65 | 4.22 | 1.36 |
| **3** | 19.74 | 4.47 | 1.17 |
| **4** | 16.70 | 3.40 | 1.10 |
| **All** | 17.39 | 3.34 | 1.11 |

***Fig. 33*** *SE-ResNet-46 test errors with different SE stages implementations*

Notice that stages 2 and 3 implementation did not improve the performance of the raw ResNet model. Only the last stage improved and even surpassed the SE-ResNet-18 model with SE blocks at all stages.

The bulk of improvement in the SE-Resnet model can thus mostly be attributed to the last stages.

### 5.1.5 Activation weights analysis

We used the same methodology as the paper to get the activation distributions across all SE blocks. 50 samples of each class were used to calculate the distribution of the filter rescaling values (**Fig. 34**).

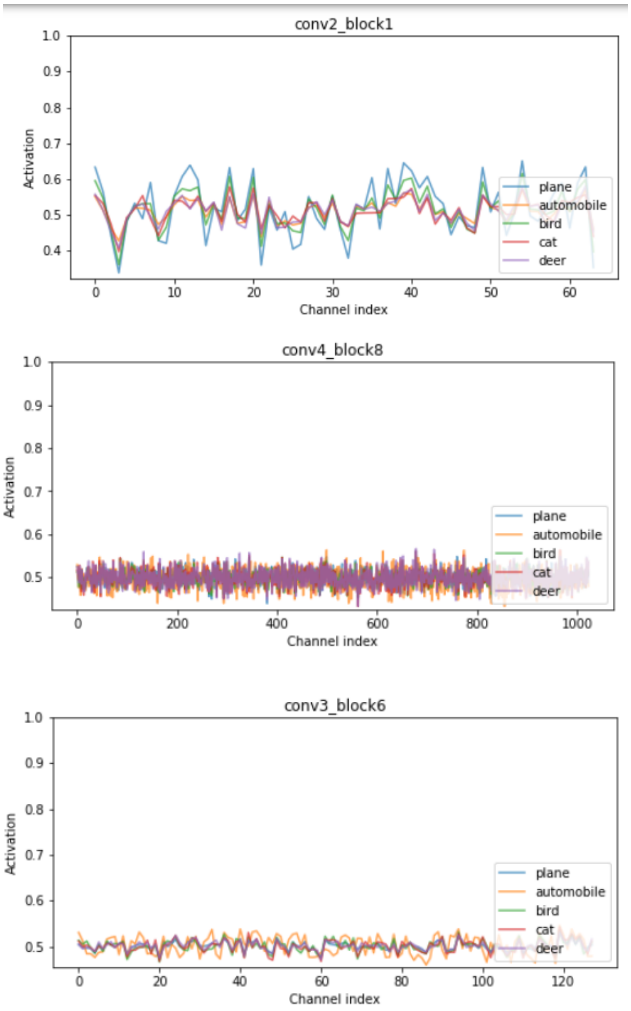



***Fig. 34*** *Activation values for each channel. Location in the ResNet-46 model is given by SE_stageID_blockID*

We can see that in earlier stages, rescaling values have a broader range. It is also easier to distinguish between the classes.

Additionally, we noticed that the standard deviation increased with the block id inside a given stage (**Fig. 35**):

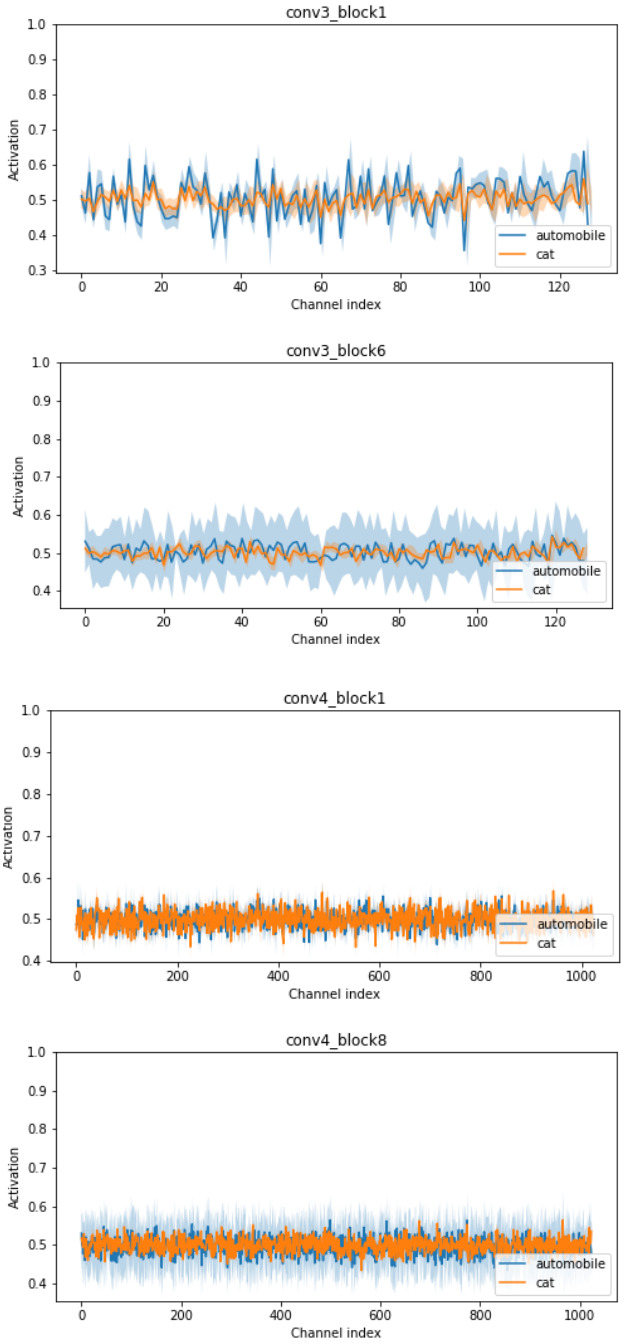



***Fig. 35*** *Activation values and standard deviation for each channel. Location in the ResNet-50 model is given by SE_stageID_blockID*

This indicates that the SE block has a dynamic behavior, especially at the end of a stage with a more diverse representation of each class. The standard deviation also changes for each class showing that SE blocks behave differently for each class.

### 5.1.6 Inference time

Inference times (**Fig. 36**) were measured on the Tiny ImageNet 9832 testing images to give an average inference time per sample.

| Model | Inference time per sample (ms) | Difference (%) |
|---|---|---|
| **ResNet18** | 0.29 | 10 |
| **SE-ResNet18** | 0.32 | |
| **ResNet34** | 0.48 | 6.3 |
| **SE-ResNet34** | 0.51 | |
| **ResNet50** | 0.57 | 16 |
| **SE-ResNet50** | 0.66 | |

***Fig. 36*** *Inference times for (64,64,3) images*

Note that inference time is hard to measure because we cannot control the way Tensorflow interacts with the GPU. Loading the samples into the GPU also takes time and is limited by the PCIe bus bandwidth which is not dependent on the model.

However, we did see an average increase of about 10% in inference time with the introduction of SE blocks.

## 5.2 Comparison of the Results Between the Original Paper and Students' Project

Because we did not use the same models, or data sets in the case of Tiny ImageNet, **no direct comparison can be made**. However we can still discuss the general results that were observed.

The changes that SE blocks added also matched the paper: it resulted in 2.5 million more parameters to our ResNet-50 as reported by the authors.

Similarly to the paper, we did see consistent improvements in testing accuracy with the addition of SE blocks (**Fig. 4, Fig. 28**). Our gains were bigger than the reported results of the paper, due the lower capacities of our models. The added overhead due to lower accuracies meant that we had to deal with variability between identical training. It did not impact the accuracy results but may have worsened the quality of the other analysis (ablation, stage, ratio).

Our ratio study (**Fig. 37**) came to the same conclusion as the paper: all ratio values yield improvements over the standard ResNet model. And higher ratios (16) give a good tradeoff between a good accuracy and increase of the number of parameters.

| Ratio | Params | top-1 err. | Params (paper) | top-1 err. (paper) |
|---|---|---|---|---|
| **ResNet** | $11 \times 10^6$ | 19.30 | $25.6 \times 10^6$ | 23.30 |
| **1** | $28 \times 10^6$ | 17.52 | x | x |
| **2** | $19.5 \times 10^6$ | 17.33 | $45.7 \times 10^6$ | 22.29 |
| **4** | $15.2 \times 10^6$ | 18.43 | $35.7 \times 10^6$ | 22.25 |
| **8** | $13.1 \times 10^6$ | 17.68 | $30.7 \times 10^6$ | 22.26 |
| **16** | $12.1 \times 10^6$ | 17.39 | $28.1 \times 10^6$ | 22.28 |
| **32** | $11.5 \times 10^6$ | 17.94 | $26.9 \times 10^6$ | 22.72 |

***Fig. 37*** *Ratio tests, paper used ResNet-50 and ImageNet. We used ResNet-46 and CIFAR-10*

The ablation study (**Fig. 38**) however did not match the paper's results with SE-POST, SE-PRE, and SE-Identity implementation giving worse results than the ResNet model.

| | top-1 err. | top-5 err. | top-1 err. paper | top-5 err paper |
|---|---|---|---|---|
| **Standard SE** | 17.39 | 1.11 | 22.28 | 6.03 |
| **SE-POST** | 21.08 | 1.11 | 22.78 | 6.35 |
| **SE-PRE** | 20.02 | 1.96 | 22.23 | 6.00 |
| **SE-Identity** | 43.78 | 6.03 | 22.20 | 6.15 |
| **Resnet** | 19.30 | 1.25 | 23.30 | 6.55 |

***Fig. 38*** *SE integration tests, paper used ResNet-50 and ImageNet. We used ResNet-46 and CIFAR-10*

This may have been corrected with a change in the training procedure, and hyper-parameters, however the paper did not mention any other changes than the SE block integration.

The SE-Identity performed the worst, while the paper mentioned that the SE-POST was the one that gave the least improvement.

The stage integration (**Fig. 39**) study also did not give satisfactory results as stage 2 and stage 3 integrations gave worse results than the regular ResNet model. While the different ResNet architectures of the paper had one more stage, they all gave improvements in accuracy. The only matching conclusion is that the deeper stages have more improvement potential with the addition of SE blocks.

| Stage | **top-1 err.** | **top-5 err.** | **top-1 err. paper** | **top-5 err. paper** |
|---|---|---|---|---|
| **2** | 19.65 | 1.36 | 23.03 | 6.48 |
| **3** | 19.74 | 1.17 | 23.04 | 6.32 |
| **4** | 16.70 | 1.10 | 22.68 | 6.22 |
| **All** | 17.39 | 1.11 | 22.28 | 6.03 |
| **Resnet** | 19.30 | 1.25 | 23.30 | 6.55 |

***Fig. 39*** *Stages tests, paper used ResNet-50 and ImageNet. We used ResNet-46 and CIFAR-10*

The activation distribution analysis also gave slightly different results. This is probably due to the different model used that had one less stage. We did not see that activations become class specific at deeper depths (**Fig. 10** and **Fig. 34**). We did however notice a higher standard deviation as the block id increases, just like in the paper (**Fig. 11** and **Fig. 35**). This indicates the dynamic nature of SE blocks that adapts differently to each image.

| | **ResNet (ms)** | **SE-ResNet (ms)** | **Difference %** |
|---|---|---|---|
| **Paper (Resnet-50)** | 0.74 | 0.82 | 11 |
| **ResNet-18** | 0.29 | 0.32 | 10 |
| **ResNet-34** | 0.48 | 0.51 | 6.3 |
| **ResNet-50** | 0.57 | 0.66 | 16 |

***Fig. 40*** *Inference times in milliseconds, paper used ImageNet. We used Tiny ImageNet with lower image resolution*

Our inference times (**Fig. 40**) matched the results of the paper with an average increase of 10%. The decrease in inference time for the same ResNet-50 model comes from the lower image resolution of Tiny ImageNet: (64,64,3) instead of (224,224,3).

## 5.3 Discussion of Insights Gained

We learned that small changes to existing network architectures can lead to measurable accuracy improvements. Implementing modern models from scratch allowed us to better understand how they work.

We had trouble choosing the right hyper-parameters, but the paper's setup was a good starting point. The most difficult part was to find a good initial learning rate and learning rate reduction strategy. We had trouble with the CIFAR data as proper convergence was challenging with existing ResNet models. Reducing the learning rate and introducing the custom ResNet-46 model allowed us to overcome this difficulty.

Carefully choosing the appropriate models and tests was also crucial as the high training time did not allow us to go back to our initial choices in order to have consistent comparisons across models and tests. If we could rerun the tests, we would likely add weight decay and more data augmentation, especially for the Tiny ImageNet data set that suffered from overfitting.

It has been learnt that even if the SGD optimizer had a lower convergence speed than other adaptive methods such as Adam, it still yields better final accuracy towards the end. Reducing the learning rate after the training reached stability allowed us to have better results.

## 6. Future Work

In the future we would like to study the SE implementation on other types of networks, such as mobile networks. As mentioned before, if the Tiny ImageNet were to be reused, adding weight decay, regularization, and more aggressive data augmentation would be a good idea.

The paper also conducted tests on other datasets such as the COCO or Places365 ones. A promising pathway would be to train our models on those sets.

## 7. Conclusion

This report summarizes the key points of the original paper [6]. It also aims to reproduce the tests and results obtained by the authors.

Adding SE blocks to existing convolutional neural networks such as ResNets or Inception networks produced consistent improvements in accuracy, while only adding about 10% parameters. This can be explained by the fact that SE-blocks improve channel interdependencies of convolutional layers.

We were able to verify the accuracy improvement over the models and data sets that we tested. However because of lower computational capabilities, the changes made to the networks and data meant that we had slightly different interpretations on certain parameters such as the effect of the location and integration method of SE layers.

We did verify the effect of the ratio parameter and the activation distribution per channels, partially matching the paper's conclusions.

Overall, the testing and analysis studies revealed some minor improvements that could be made in the future regarding the overfitting of the Tiny ImageNet data set for example. Also other models and data sources could be investigated in future works.

## 8. Acknowledgement

The implementation of the models was possible thanks to the papers that explained them [2][3][8] and thanks to the teaching of **Prof. Z. Kostic**.

The Tensorflow and Keras documentation was useful for the training functions, callbacks, and data augmentations.

## 9. References

[1] Anh-Vu Nguyen, Antonin Vidon, Wael Boukhobza (2021). Squeeze-and-Excitation Networks [code] https://github.com/ecbme4040/e4040-2021fall-project-fren-an3078-wab2138-av3023

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in CVPR, 2016.

[3] C. Szegedy et al., Going Deeper with Convolutions. 2014.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in ICCV, 2015.

[5] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep
network training by reducing internal covariate shift," in ICML,
2015.

[6] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, Squeeze-and-Excitation Networks. 2019.

[7] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in CVPR, 2017.

[8] S. Hitawala, Evaluating ResNeXt Model Architecture for Image Classification, CoRR. abs/1805.08700 (2018).

[9] Krizhevsky, Alex, Vinod Nair, and Geoffrey Hinton. "The CIFAR-10 dataset." online: http://www.cs.toronto.edu/kriz/cifar. html (2014).

[10] Krizhevsky, Alex, Vinod Nair, and Geoffrey Hinton. "The CIFAR-100 dataset." online: http://www.cs.toronto.edu/kriz/cifar. html (2014).

[11] Jiayu Wu, Qixiang Zhang, and Guoxi Xu. Tiny imageNet challenge. Technical Report, 2017

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," International Journal of Computer Vision, 2015.

[13] G. Pleiss, T. Zhang, E. R. Elenberg, en K. Q. Weinberger, "Identifying Mislabeled Data using the Area Under the Margin Ranking", arXiv [cs.LG]. 2020.

[14] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna, "Rethinking the Inception Architecture for Computer Vision", arXiv [cs.CV]. 2015.

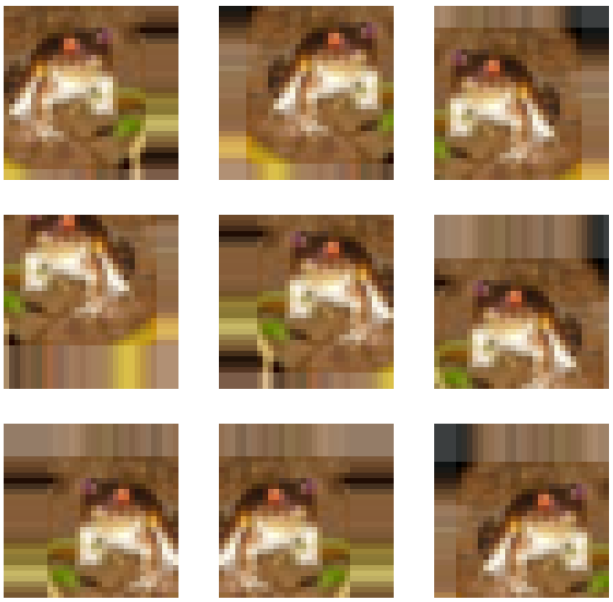
## 10. Appendix

### 10.1 Individual Student Contributions in Fractions

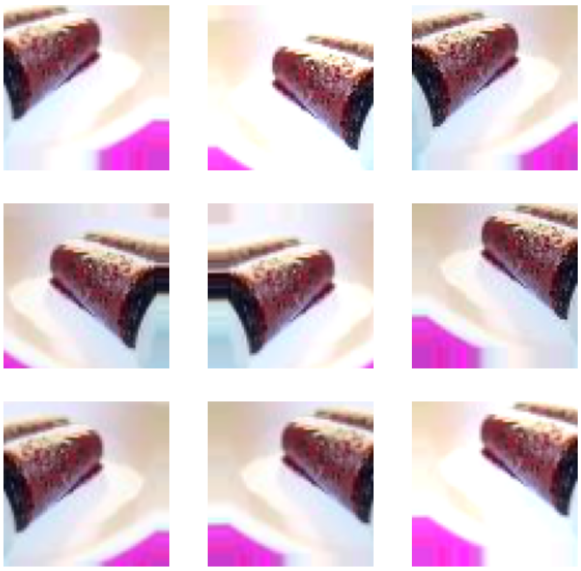
| | an3078 | av3023 | wab2138 |
|---|---|---|---|
| Last Name | Nguyen | Vidon | Boukhobza |
| Fraction of (useful) total contribution | 1/3 | 1/3 | 1/3 |
| What I did 1 | Reading the paper<br>Choosing datasets<br>Modeling | | |
| What I did 2 | Built ResNet models and SE variants | Built ResNeXt models and SE variants | Built inception models and SE variants |
| What I did 3 | Training and testing | | |

| What I did 4 | Wrote the report |
|---|---|

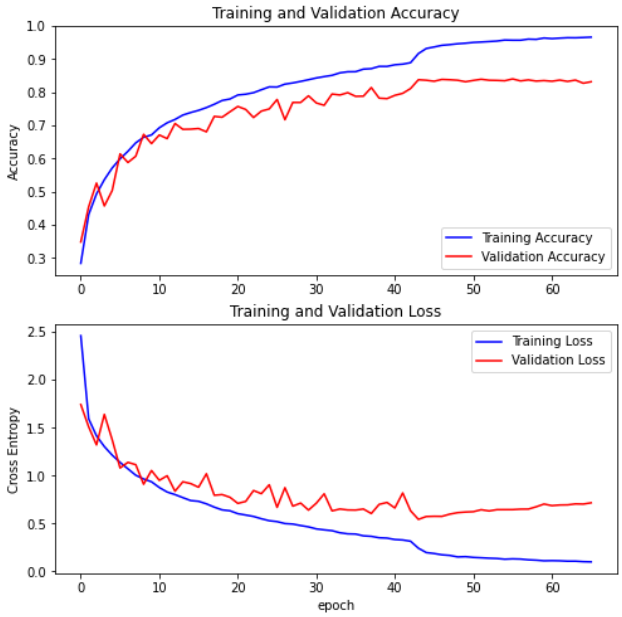
### 10.2 Support Material



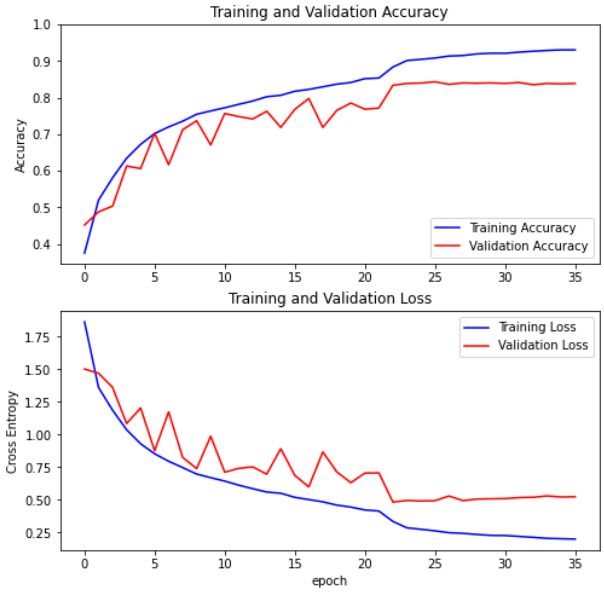
*Data augmentations for CIFAR-10 and 100.*



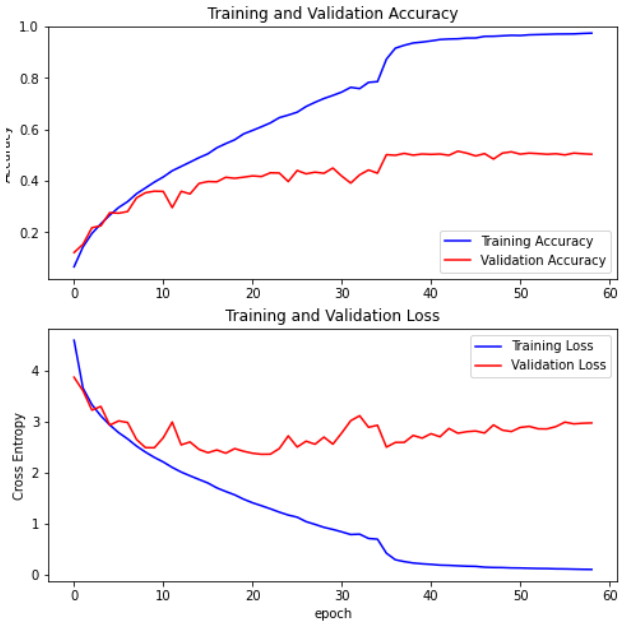
*Data augmentations for Tiny ImageNet*



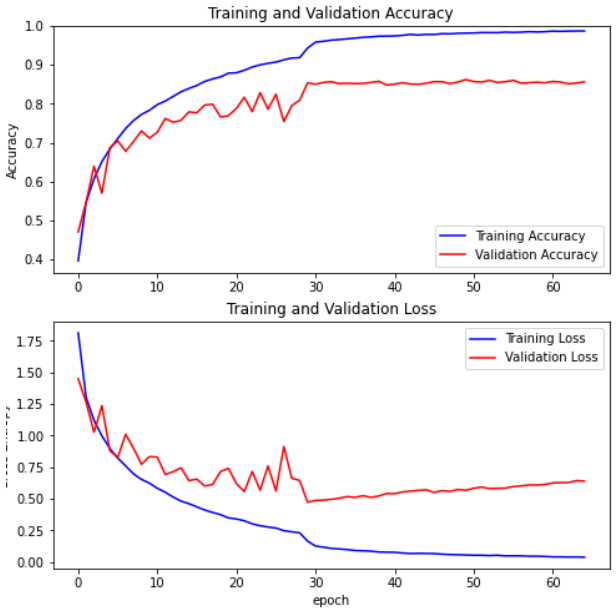

*Training metrics for the ResNet-46 model with the CIFAR-10 data set - best model saved at epoch 55*




*Training metrics for the SE-ResNet-46 model with the CIFAR-10 data set - best model saved at epoch 25*

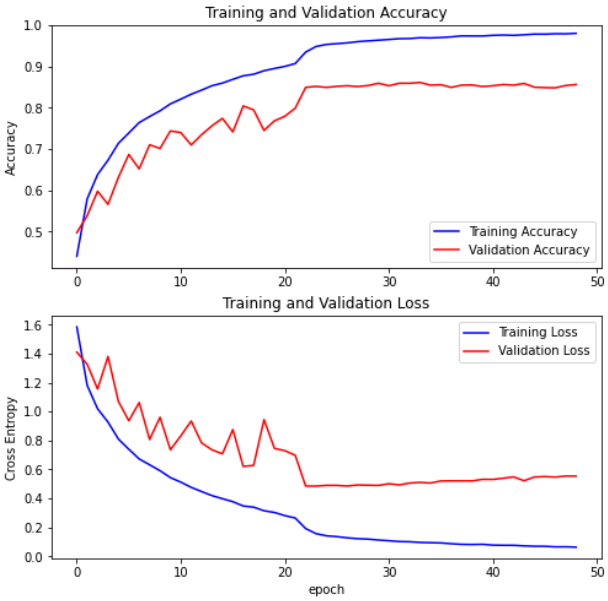



*Training metrics for the ResNet-46 model with the CIFAR-100 dataset -best model saved at epoch 43*

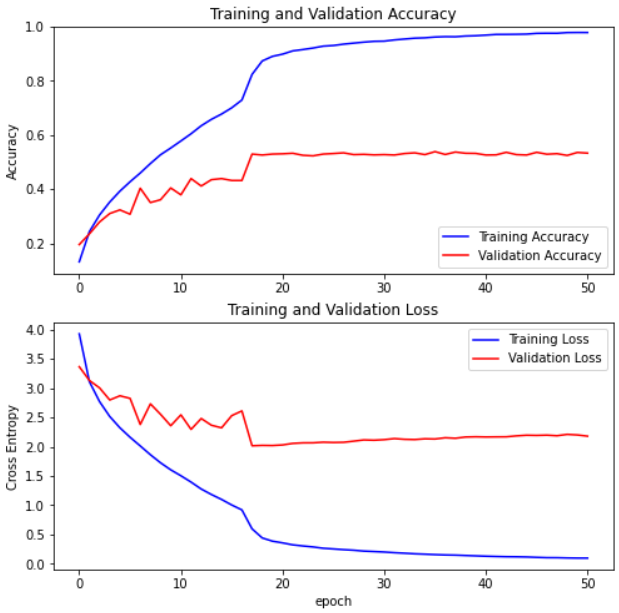



*Training metrics for the ResNeXt-29 model with the CIFAR-10 data set - best model saved at epoch 49*

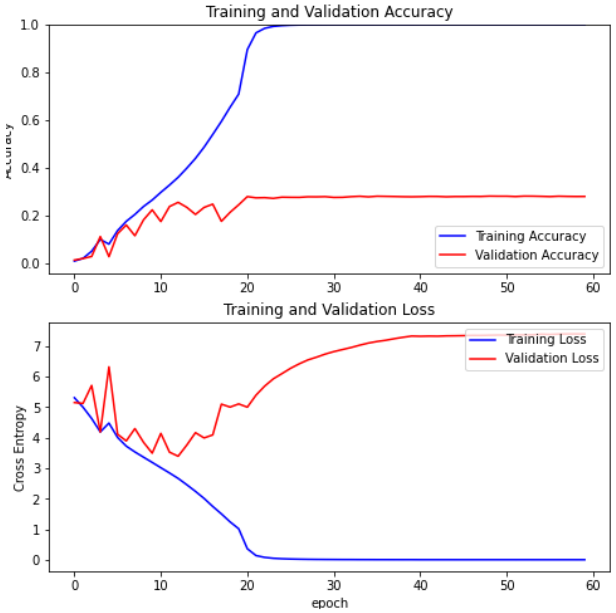

*Training metrics for the SE-ResNet-46 model with the CIFAR-100 dataset - best model saved at epoch 47*



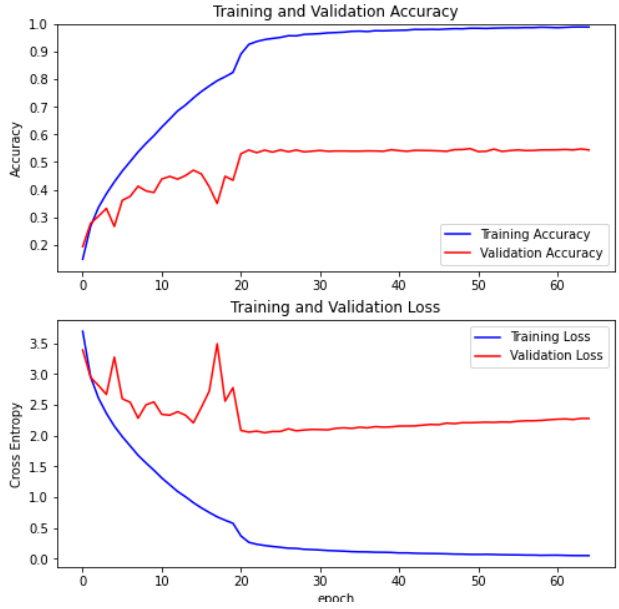

*Training metrics for the SE-ResNeXt-29 model with the CIFAR-10 data set - best model saved at epoch 34*



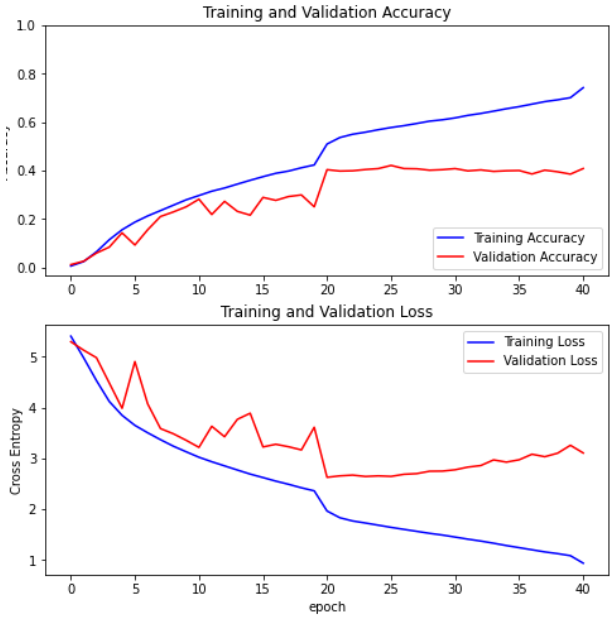

*Training metrics for the ResNeXt-29 model with the CIFAR-100 data set - best model saved at epoch 36*



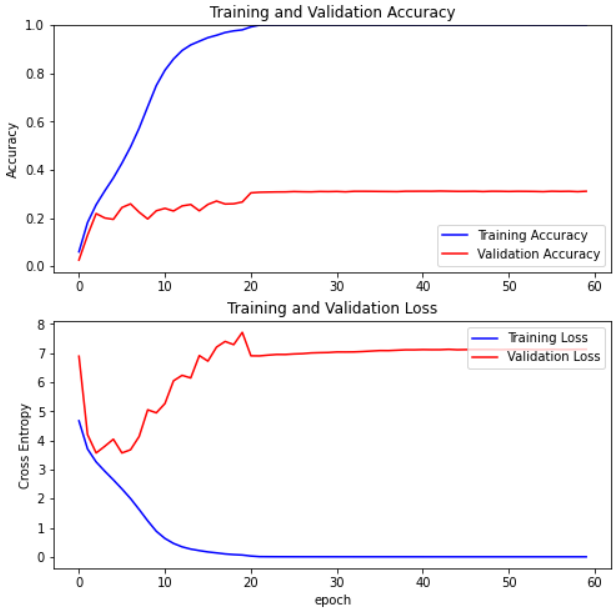

*Training metrics for the ResNet-18 model with the Tiny ImageNet dataset without data augmentation - best model saved at epoch 36*



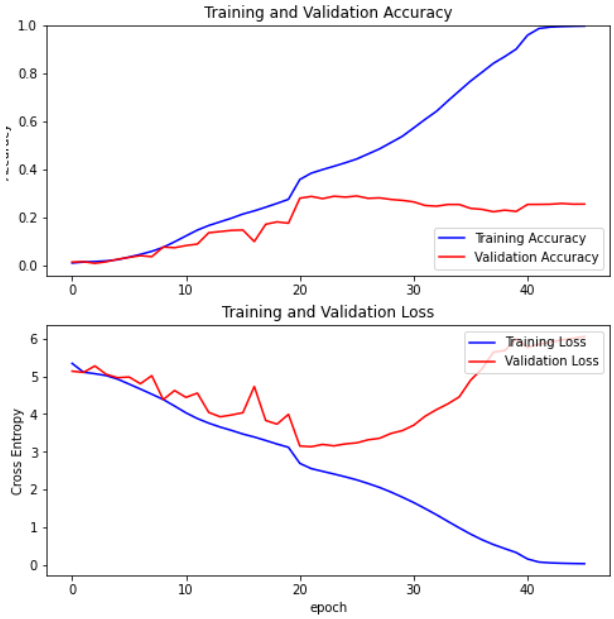

*Training metrics for the SE-ResNeXt-29 model with the CIFAR-100 data set - best model saved at epoch 50*



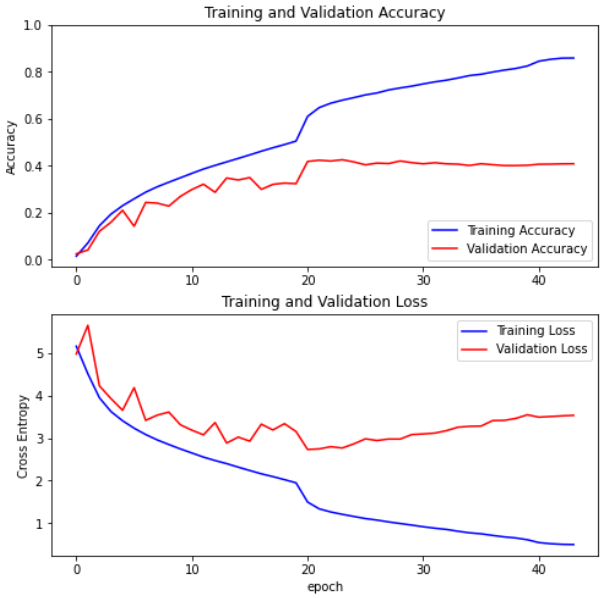

*Training metrics for the ResNet-18 model with the Tiny ImageNet dataset with data augmentation - best model saved at epoch 26*



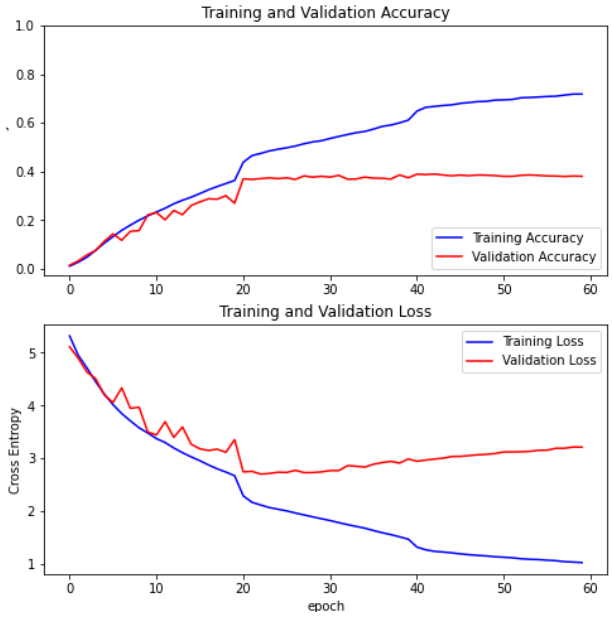

*Training metrics for the SE-ResNet-18 model with the Tiny ImageNet dataset without data augmentation - best model saved at epoch 43*



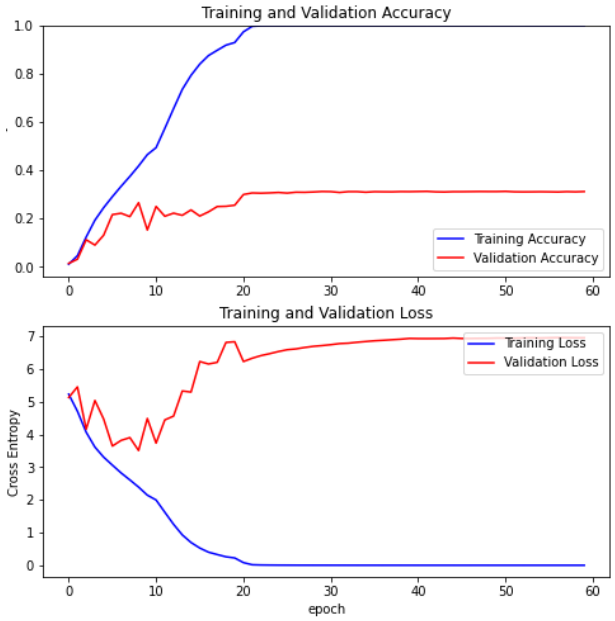

*Training metrics for the ResNet-34 model with the Tiny ImageNet dataset without data augmentation - best model saved at epoch 26*



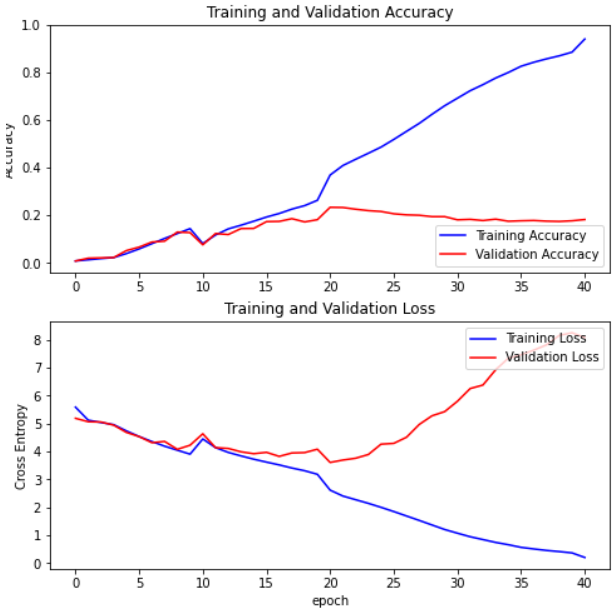

*Training metrics for the SE-ResNet-18 model with the Tiny ImageNet dataset with data augmentation - best model saved at epoch 24*



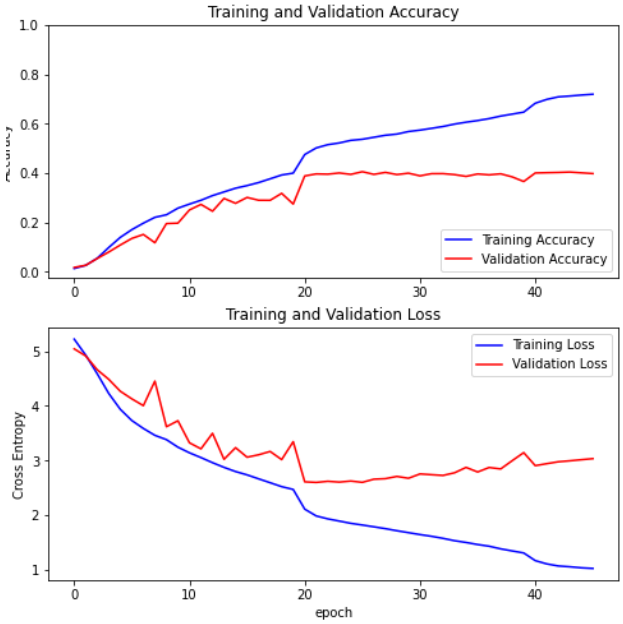

*Training metrics for the ResNet-34 model with the Tiny ImageNet dataset with data augmentation - best model saved at epoch 43*



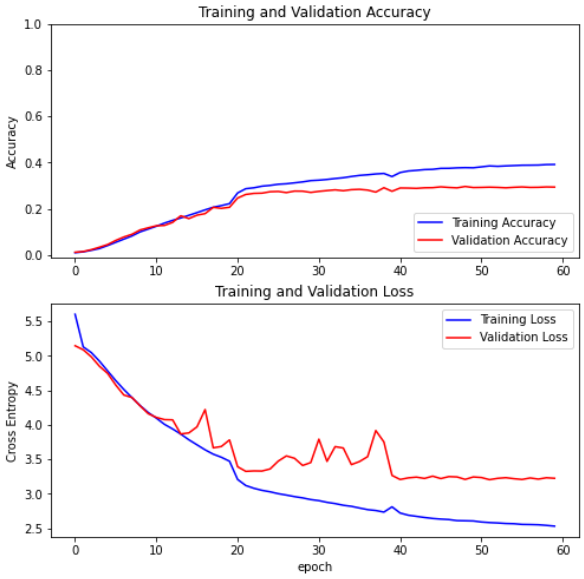

*Training metrics for the SE-ResNet-34 model with the Tiny ImageNet dataset without data augmentation - best model saved at epoch 42*



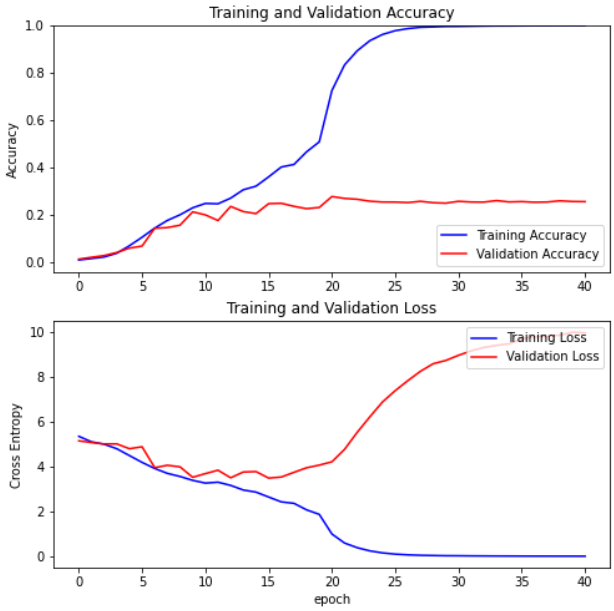

*Training metrics for the ResNet-50 model with the Tiny ImageNet dataset without data augmentation - best model saved at epoch 21*



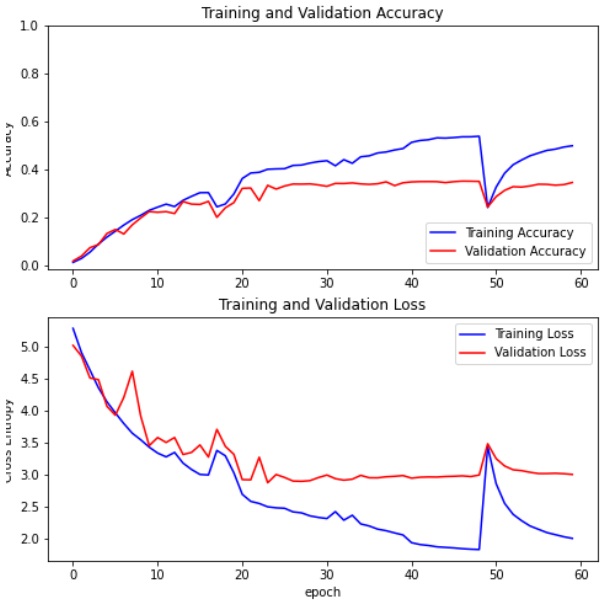

*Training metrics for the SE-ResNet-34 model with the Tiny ImageNet dataset with data augmentation - best model saved at epoch 26*



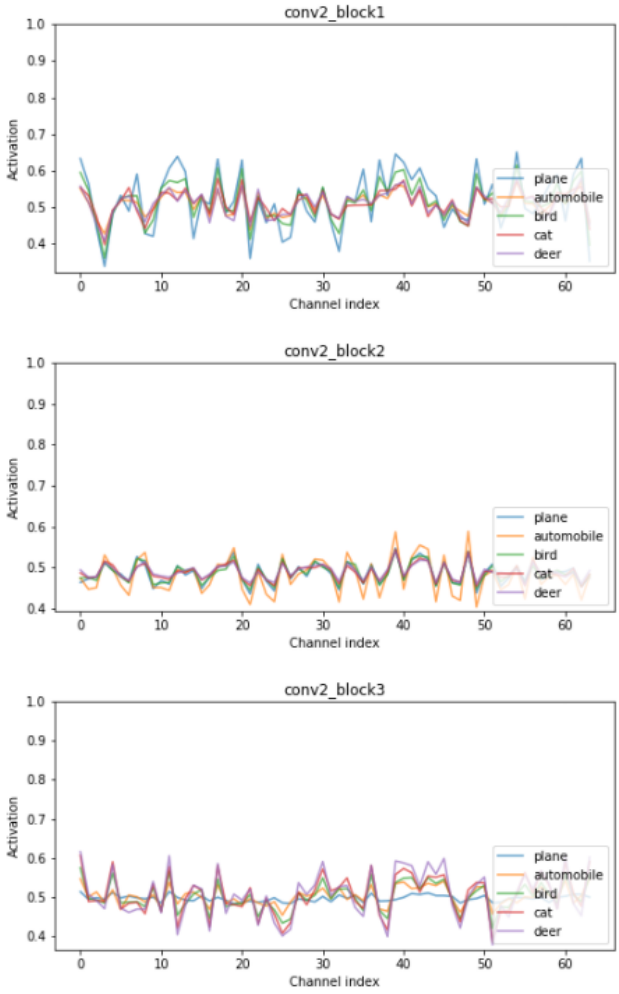

*Training metrics for the ResNet-50 model with the Tiny ImageNet dataset with data augmentation - best model saved at epoch 49*



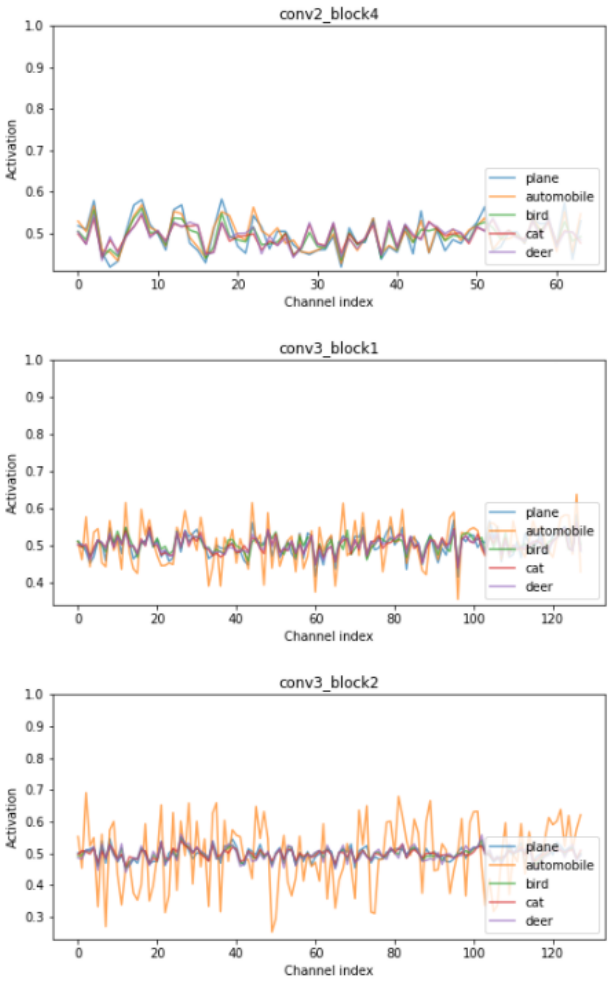

*Training metrics for the SE-ResNet-50 model with the Tiny ImageNet dataset without data augmentation - best model saved at epoch 2*



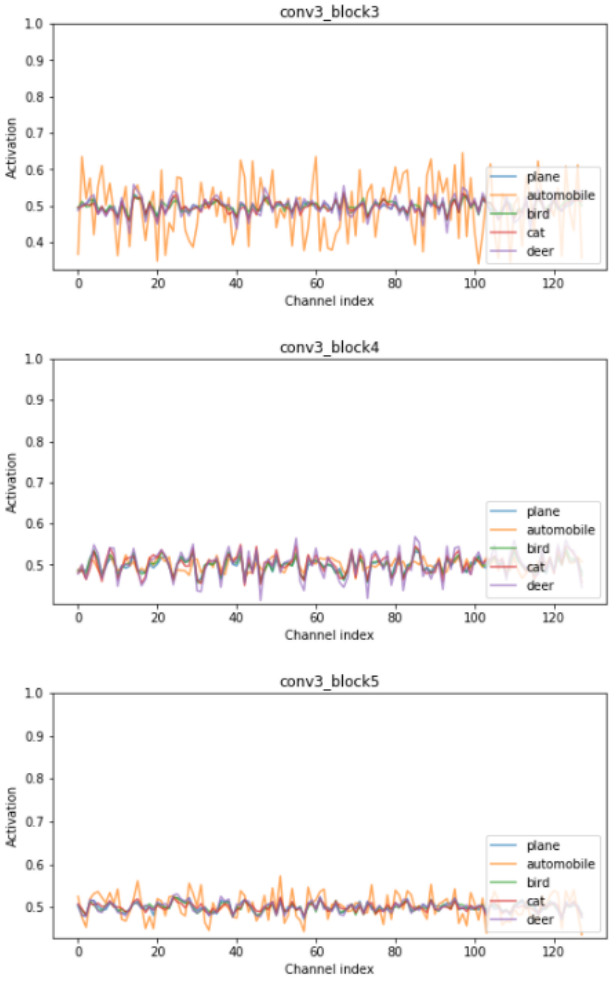

*Training metrics for the SE-ResNet-50 model with the Tiny ImageNet dataset with data augmentation - best model saved at epoch 47*



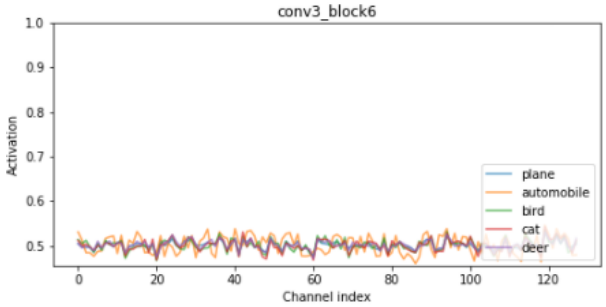

*Activation channel distribution for 5 classes. SE-block location is given by convSTAGE_blockID (1)*

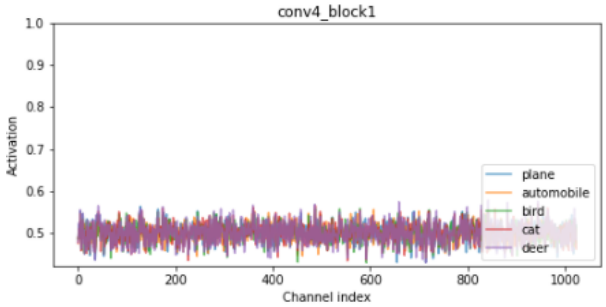


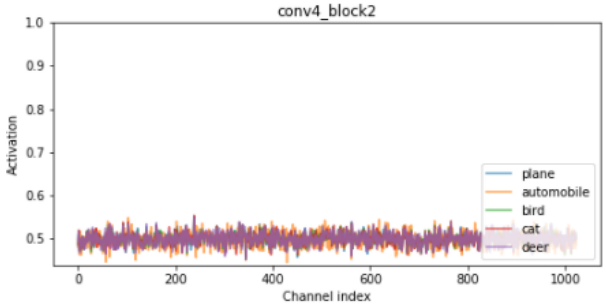

*Activation channel distribution for 5 classes.*
*SE-block location is given by convSTAGE_blockID (2)*



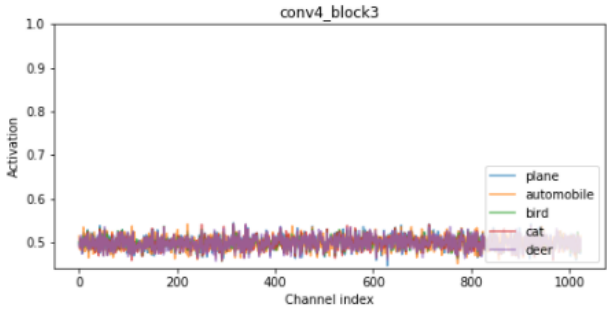

*Activation channel distribution for 5 classes.*
*SE-block location is given by convSTAGE_blockID (3)*

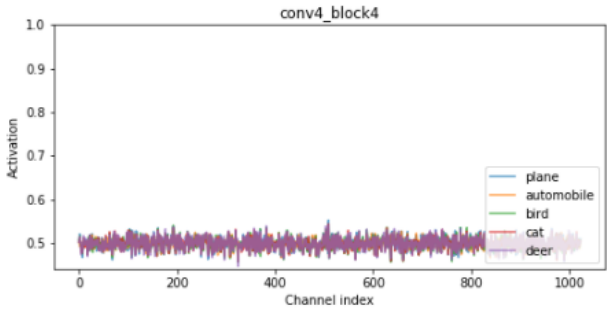


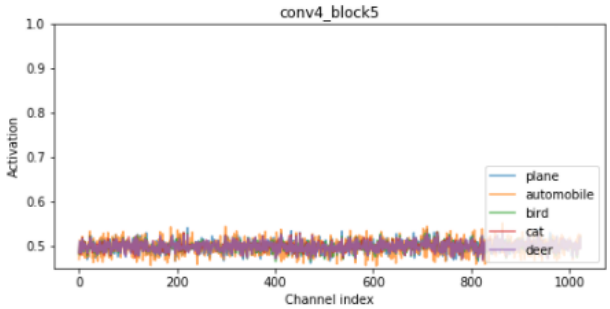



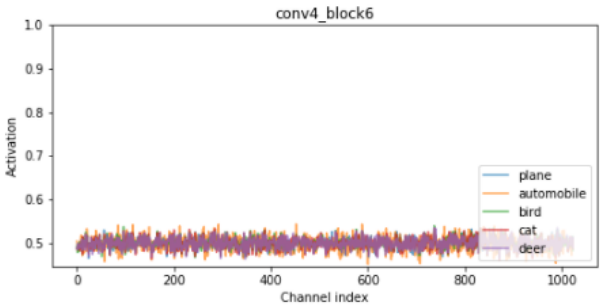

*Activation channel distribution for 5 classes.*
*SE-block location is given by convSTAGE_blockID (4)*

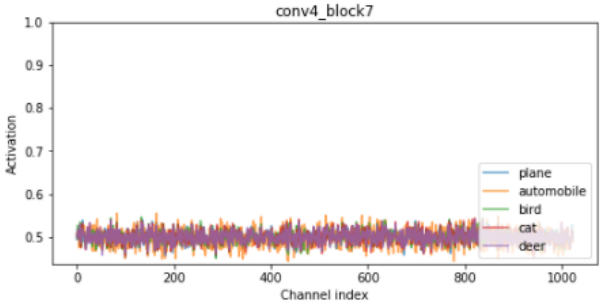


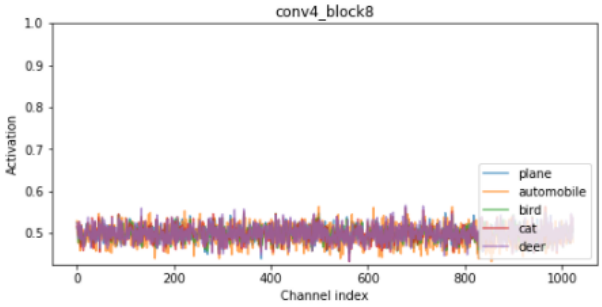



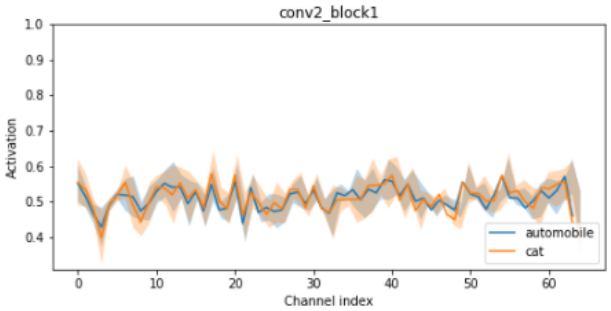

*Activation channel distribution for 5 classes.*
*SE-block location is given by convSTAGE_blockID (5)*

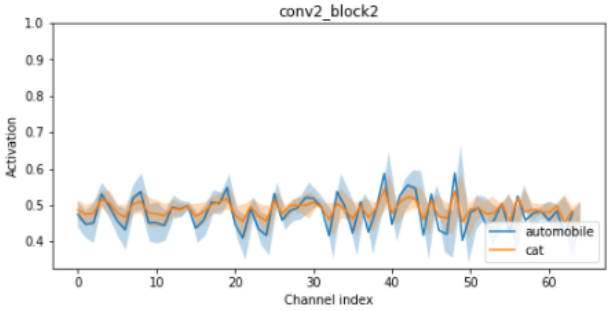


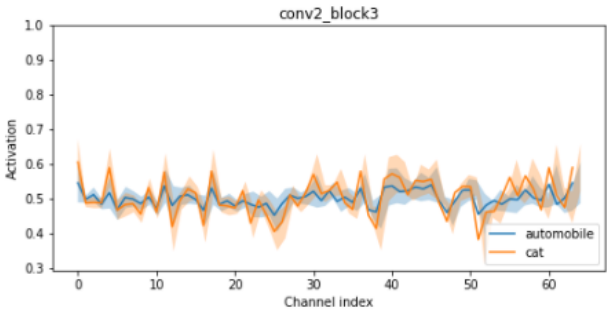



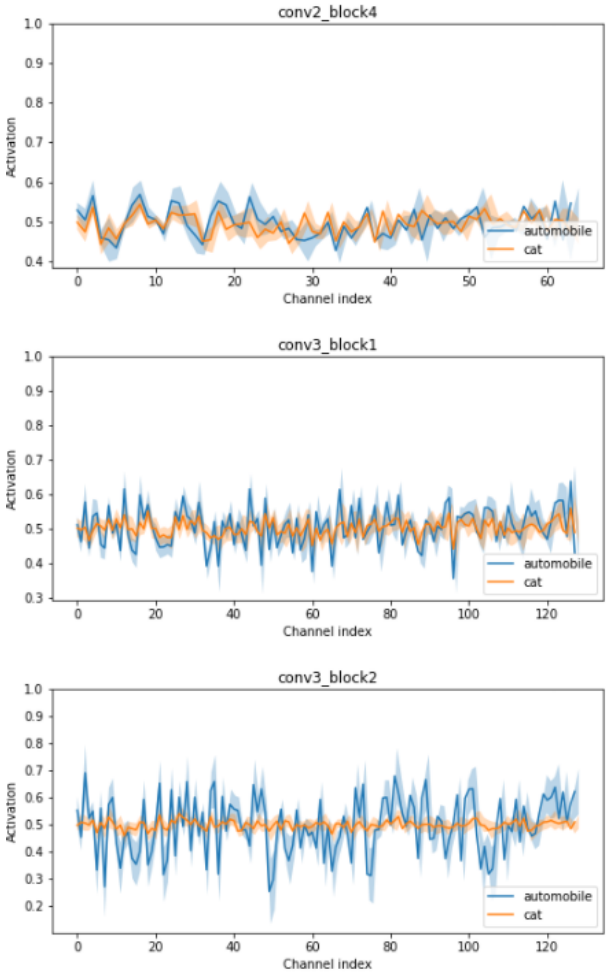

*Activation channel distribution for 5 classes. SE-block location is given by convSTAGE_blockID (6)*



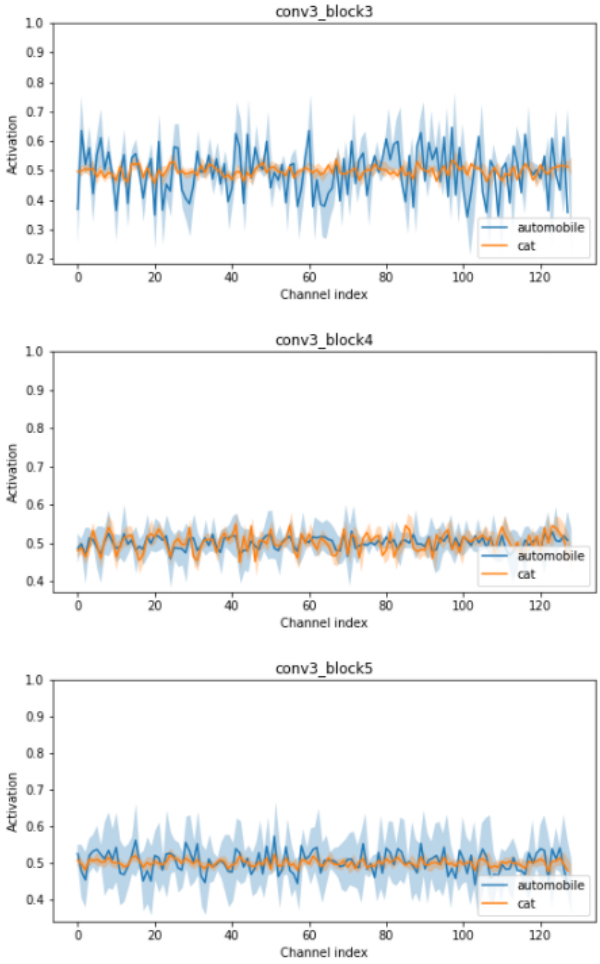



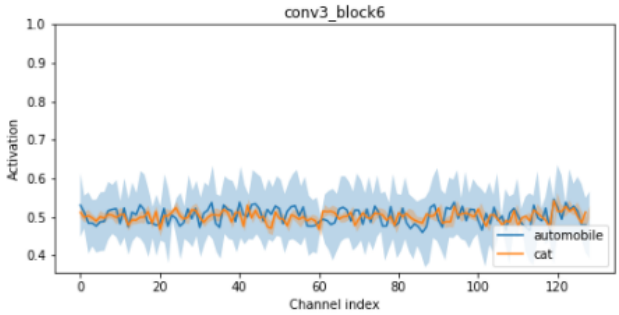

*Activation channel distribution and standard deviation for 2 classes. SE-block location is given by convSTAGE_blockID (1)*

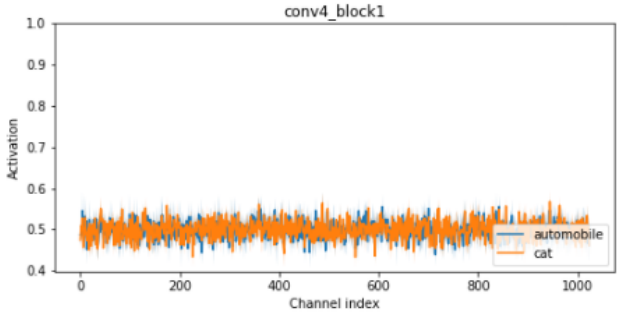


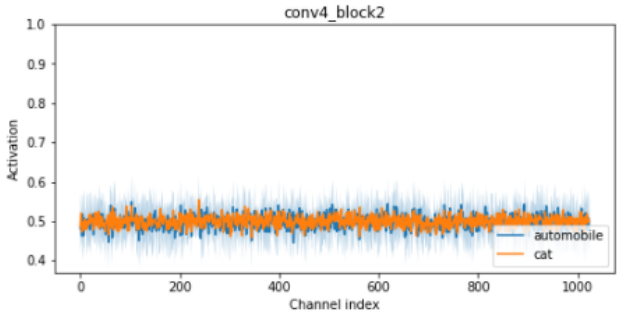

*Activation channel distribution and standard deviation for 2 classes. SE-block location is given by convSTAGE_blockID (2)*



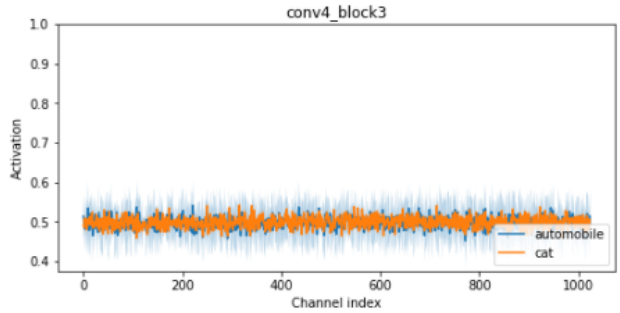

*Activation channel distribution and standard deviation for 2 classes. SE-block location is given by convSTAGE_blockID (3)*

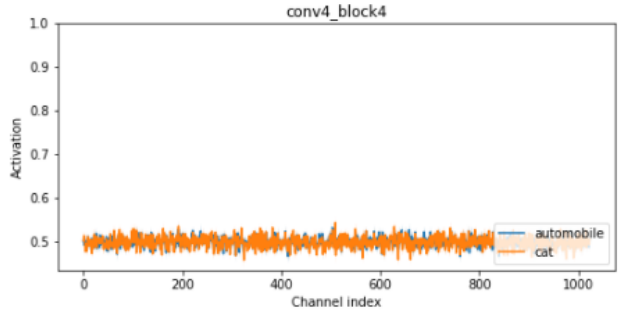


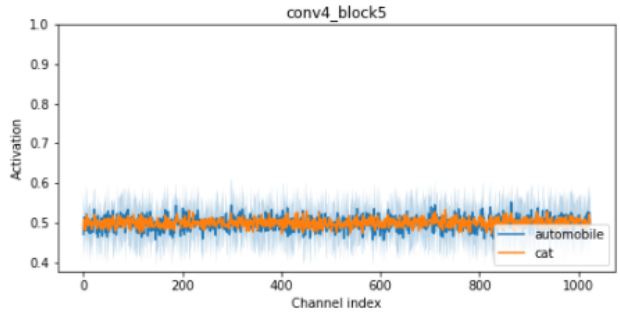



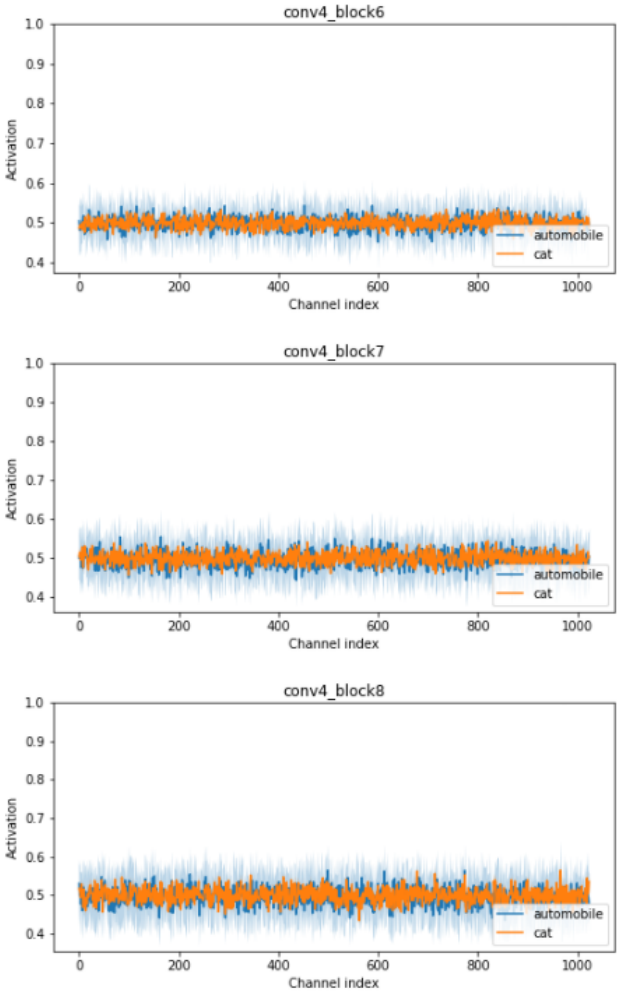

*Activation channel distribution and standard deviation for 2 classes. SE-block location is given by convSTAGE_blockID (4)*




*Activation channel distribution and standard deviation for 2 classes. SE-block location is given by convSTAGE_blockID (5)*




*Activation channel distribution and standard deviation for 2 classes. SE-block location is given by convSTAGE_blockID (6)*

.